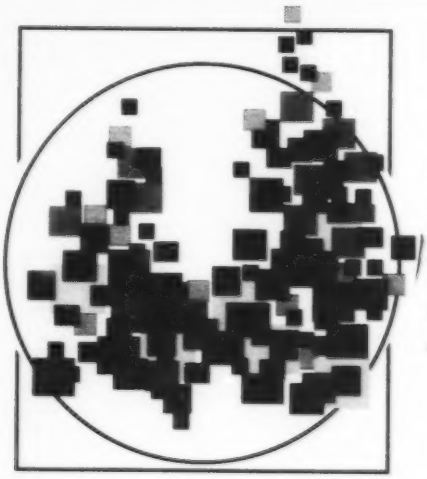

CENTRE FOR
SYSTEMS
SCIENCE

Simon Fraser University, Burnaby, B.C., Canada V5A 1S6 / (604) 291-4704

A Tree Unification Grammar-Based Natural Language Processor

Fred Popowich

CSS/-IS TR 89-08

December 1989

A Tree Unification Grammar-Based Natural Language Processor

Fred Popowich

CSS-IS TR 89-08

Abstract

A natural language processor is described that provides translations of input sentences into a semantic notation. These sentences may contain reflexive pronouns, proper names, indefinite noun phrases, possessives, picture-nouns, relative clauses, ditransitive verbs and controlled complements. Syntactic, semantic and anaphoric processing are all done simultaneously. The processor consists of a Tree Unification Grammar containing a single grammar rule, and a general Tree Unification Grammar chart parser.

List of Figures

Figure 2-1: Structure of Category Names. 4
Figure 2-2: Lexical Entries. 5
Figure 2-3: FA Specification. 6
Figure 2-4: FA Structure. 7
Figure 2-5: Templates for Phonological Information. 8
Figure 2-6: Lexical Rule for Null Subject Instantiation. 8
Figure 3-1: Templates for Syntactic Information. 9
Figure 3-2: Templates for Semantic Information. 10
Figure 3-3: Templates for Reflexive Information. 10
Figure 3-4: Templates for Noun Phrases. 11
Figure 3-5: Lexical Entry for *gives*. 12
Figure 3-6: Template Expansion. 13
Figure 3-7: Lexical Entry for *herself*. 13
Figure 3-8: Lexical Entry for *of*. 13
Figure 3-9: FA Specification for a Picture-Noun. 14
Figure 3-10: FA Specification for the Indefinite Article. 15
Figure 3-11: Lexical Entry for *loves*. 15
Figure 3-12: FA Specification for a Verb Phrase. 16
Figure 3-13: Lexical Entry for Possessive. 16
Figure 3-14: Possessive with Picture-Noun. 17
Figure 3-15: Lexical Entry for *persuades*. 19
Figure 3-16: Lexical Entry for Verb in Controlled Complement. 20
Figure 3-17: Lexical Entry for Auxiliary Verb *to*. 20
Figure 3-18: Lexical Entry for Restrictive Relative Pronoun (preliminary formulation). 24
Figure 3-19: Lexical Entry for a Trace. 24
Figure 3-20: Lexical Entry for Restrictive Relative Pronoun. 26
Figure 3-21: Reflexive in Relative Head. 27
Figure 4-1: Initial Chart. 28
Figure 4-2: Rule Invocation. 29
Figure 4-3: Initial Application of Fundamental Rule. 29
Figure 4-4: Final Application of Fundamental Rule. 30
Figure 4-5: Term Representation of a Tree. 30
Figure 4-6: Lexical Entry for *herself*. 31
Figure 4-7: Term Representation of an FA Specification. 31
Figure 4-8: The Chart Parser. 32
Figure 4-9: Fundamental Rule. 33
Figure 4-10: Rule Invocation. 36
Figure 4-11: Edge with Constraints. 37
Figure 4-12: Lexical Entry for a Single Sentence Discourse. 38
Figure 5-1: Trace of Successful Parse. 38
Figure 5-2: Analysis of an Ambiguous Sentence. 39

1. Introduction

Tree Unification Grammar (TUG) (Popowich, 1988, Popowich, 1989a) is a member of the family of unification-based formalisms (Shieber, 1986) which includes Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1987), Unification Categorial Grammar (UCG) (Zeevat, Klein and Calder, 1987, Calder, Klein, and Zeevat, 1988) and Categorial Unification Grammar (CUG) (Uszkoreit, 1986). Associated with the development of unification-based formalisms was a shift towards greater lexicalism: more complex lexical entries and fewer grammar rules. TUG takes this lexical shift to the extreme with its proposal of only a single grammar rule. Another distinguishing feature of TUG is that partial specifications of binary trees are the *grammar structures* that are manipulated by this single grammar rule.

In this paper, a grammar for a small subset of English will be introduced that consists of a single rule along with a selection of lexical entries that illustrates some interesting properties of TUG. The TUG treatment of reflexivization, control and unbounded dependencies will be highlighted as will be the highly structured TUG lexicon. A chart parser for TUG that takes advantage of characteristic properties of the formalism will then be introduced. This chart parser forms the basis of a system which can give semantic representations for input sentences and which can use semantic and anaphoric constraints, as well as syntactic constraints, to restrict possible analyses during processing.

2. Tree Unification Grammar

The TUG formalism developed out of research into a unification-based treatment of anaphora (in particular, reflexivization) within the UCG framework. The goal was to allow constraints on anaphora to play a role in restricting possible analyses of linguistic expressions. Consider the sentence shown in (1) in which the anaphor *herself* cannot take *John* (or any other noun phrase in the sentence) as its antecedent.

(1) *John's portrait of herself was hidden in the attic.

Some linguistic and computational linguistic treatments of anaphora state constraints on anaphora in terms of principles or restrictions on grammar structures (e.g., the c-command restriction (Reinhart, 1981) on syntactic structures and the f-command restriction (Kameyama, 1985) on the f-structures of Lexical Functional Grammar (Kaplan and Bresnan, 1982)) while others incorporate them into grammar rules or onto principles associated with grammar rules (Bach and Partee, 1980, Pollard and Sag, 1983, Johnson and Klein, 1986). The advantage of the latter method is that anaphora resolution (i.e., the establishment of an anaphor-antecedent relationship) can be performed during syntactic analysis. Consequently, failure of anaphora resolution can be used to block analyses — there is no need for a complete syntactic analysis of ungrammatical sentences like (1).

Unfortunately, incorporation of the constraints associated with reflexivization into UCG or HPSG rules or principles results in either an increase in the number of grammar rules (Popowich, 1988, Popowich, 1989b) or requires an associated principle for reflexivization which cannot be directly stated in the language of the formalism (Klein, 1987, Pollard and Sag, 1989). In TUG, the constraints on reflexivization

can be stated in the lexicon without requiring additional grammar rules (Popowich, 1988, Popowich, 1989a). This is a consequence of the TUG lexical entries closely resembling the derivation trees of these other formalisms. Furthermore, these constraints on reflexivization can also be used to restrict possible analyses; there will be no well-formed grammar structures for linguistic expressions in which anaphora resolution fails.

2.1. FA Structures

The basic grammar structures of TUG are binary trees called *function argument* (FA) structures. For any nonterminal node in an FA structure, one of the two branches leading from the node is designated to be its *functor*, and the other, its *argument*. Later, we shall see how this functor-argument distinction corresponds to different syntactic, semantic and anaphoric relationships.

All the nodes of the binary tree are labelled with information structures called *signs* (cf. Pollard and Sag, 1987; Calder, Klein, and Zeevat, 1988). For any subtree of an FA structure, we will refer to the sign labelling the root of the subtree as the *root-sign*. The terms *functor-sign* and *argument sign* will refer to the signs labelling the nodes accessed by the functor and argument respectively. We will also use the terms *sister-sign* and *parent-sign* to refer to the signs associated with sister and parent nodes in a subtree. A sign contains attributes for phonological, syntactic, semantic and anaphoric information. The first three of these attributes correspond to those present in the signs of HPSG and UCG. An attribute for anaphoric information, distinct from semantic information, is introduced to contain the anaphoric context of an expression.

The phonology attribute consists of a sequence of strings (intuitively corresponding to words or parts or words) separated by hyphens. This follows the convention adopted in UCG and HPSG whereby the phonology attribute simply contains orthographic information.

The syntactic information of a sign consists of *intrinsic* syntactic information associated with a linguistic expression along with *nonintrinsic* or *relational* information. For our needs, the intrinsic information will take the form of a list containing i) a category specification and ii) a case (for *nominal* categories) or form (for *verbal* categories) specification. The relational information consists of a list of signs, called the slash (Gazdar et al, 1985) list, which is used in the treatment of unbounded dependencies (cf. §3.5).

The semantic information of a sign consists of an Indexed Language (InL) formula (Calder, Klein, and Zeevat, 1988, Zeevat, 1988).[1] Many constraints on anaphora can be formulated elegantly with respect to InL formula since InL incorporates the fundamental features of a theory known for its elegant treatment of anaphoric phenomena, namely Discourse Representation Theory (DRT) (Kamp, 1981). InL formulae make use of *sorted* variables for distinguishing between different kinds of events, states and entities. For

[1]The syntax of InL as presented in this paper differs slightly from that used by Calder, Klein, and Zeevat (1988) and by Zeevat (1988). InL is discussed in detail by Popowich (1988).

example, we will use variables of the *masculine-singular-object* sort to correspond to entities "referenced" by the pronoun *himself*. Unification of variables with incompatible sorts (like masculine and feminine, or state and entity) fails, thus preventing analyses for sentences like (1). For InL formulae, we adopt the convention that variables of the *masculine-singular-object* sort will start with the letter *m*, *feminine-singular-object* with *f*, *neuter-singular-object* with *n*, *event* with *e*, and *state* with *s*. Variables for *singular-object* sort entities that are unspecified with respect to gender will start with the letters *x*, *y* and *z*. An InL formula is of the form *[a]C* where *a* is a sorted variable called the *index* of the formula and *C* is a predicate logic-like expression known as the *condition*. Since the index is similar to a discourse marker in DRT, we will often refer to indices and InL variables as discourse markers. InL formulae associated with linguistic expressions describing or modifying objects will possess an index of the appropriate sort for the object. For example, the formula corresponding to the translation of the common noun *man* will possess an index of the masculine-singular-object sort. The condition consists of a predicate name followed by an argument list where each argument is either a sorted variable or an InL formula. There are two special binary predicates, *and* and *implies*, which correspond to conjunction and DRT implication, respectively. Formulae containing these predicates are often displayed in infix notation with *and* and *implies* being replaced by a comma and an arrow respectively. This means that *[A,B]* is an abbreviation for *and(A,B)*, and *[A⇒B]* is an abbreviation for *implies(A,B)*, where *A* and *B* stand for arbitrary InL formulae. InL formulae will also be abbreviated by omitting the index when it is also the first argument of the predicate. So, *[e]love(e,x,y)* can be abbreviated as *love(e,x,y)*. Although InL does not have an explicit negation predicate or connective, negation can be defined in terms of *implies* and the necessarily false formula *[a]false(a)* as shown in (2).

(2) [a'][A ⇒ false(a)]

We will also introduce a special indexed formulae with a necessarily true condition which will be required in some of our lexical entries. The InL formula *[a]true(a)* is designated to be true for all *a*.

Anaphoric information is distinguished from semantic information on the premise that semantic information is *compositional* while anaphoric information is *environmental*. The notion of compositionality has been the subject of a great deal of discussion with respect to natural language semantics, and numerous definitions of this term have been proposed (Partee, 1984). We will use this term to reflect the fact that for any subtree of an FA structure, the semantic attribute of the root-sign will be composed of the semantic information contained in its functor-sign and argument-sign. According to this definition, our phonological information will also be compositional. Our anaphoric information will be environmental in that it can contain anaphoric information from its parent-sign and anaphoric (and even semantic) information from its sister-sign. The only anaphoric information that concerns us is the information responsible for reflexivization — information relevant for nonreflexive pronouns is discussed by Popowich (1988). This *reflexive antecedent* (R-antecedent) information consists of a list of discourse markers (cf. Johnson and Klein, 1986) corresponding to possible antecedents for reflexive pronouns (Popowich, 1989a).

All of these different kinds of information can be seen in the sign associated with the reflexive

pronoun *himself* introduced in (3).

(3) himself
[np, obj] [ ]
[m] true(m)
[m]

The syntactic information specifies that "himself" is a noun phrase (np) possessing the objective case (obj). Note that the category name *np* can be viewed as an abbreviation for a complex feature structure containing attributes *nominal*, *verbal*, and *bar level* (Chomsky, 1970). The conventional categories for

		+nominal	-nominal
+verbal	bar 1	adj	v
	bar 2	adjp	sent
-verbal	bar 1	n	p
	bar 2	np	pp

Figure 2-1: Structure of Category Names.

nouns, verbs, prepositions and adjectives can be defined in terms of these three attributes as shown in Figure 2-1. Not all categories can be defined in terms of these attributes, however. For this reason, minor categories like *det* for determiners will also be required. The second part of the syntax attribute in (3) consists of an empty list; the reflexive pronoun is not involved in any unbounded dependency constructions. The semantic attribute contains a very simple InL formula. The index of this formula is a variable of the masculine-singular-object sort, with the condition consisting of the predicate *true* applied to this variable. Since this condition is defined to be satisfiable for any variable, we can intuitively paraphrase the semantic attribute of this sign as stating that the reflexive pronoun introduces a discourse marker (of the appropriate sort) but does not impose any further restrictions on it. This semantic variable is token-identical to the one appearing in the reflexive attribute of the sign; the discourse marker associated with the reflexive pronoun is a member of the reflexive attribute. Recall that the reflexive attribute is defined to be a list of semantic variables or discourse markers corresponding to potential antecedents. For the sign introduced in (3), this list contains only one element.

2.2. FA Specifications

Partial specifications of FA structures are called FA specifications. An FA specification acts as a restriction on the structure of the binary tree which makes up an FA structure, and as a restriction on the labels contained in an FA structure. It describes a set of FA structures that are compatible with it. FA specifications are combined according to the TUG grammar rule to provide more and more information about an FA structure. As an FA specification becomes more complex, the number of FA structures described by the specification decreases. In this way, FA specifications acting as lexical entries provide restrictions on the FA structure that can be associated with a linguistic expression containing the lexical entries. The TUG grammar rule provides a means for relating these restrictions. Construction of a complex FA specification according to the grammar rule is thus a monotonic process in that restrictions

are proposed and never rescinded.

An FA specification consists of binary tree along with an *auxiliary list* consisting of pointers to different nodes in the tree. This list of pointers, which is displayed in angle brackets to the left of the tree specifies subtrees of the FA specification with which other FA specifications must be unified. As FA specifications are partial descriptions of FA structures, we will use variables whose names begin with uppercase letters or an underscore to represent the unspecified values within the signs of the FA specification. When the underscore alone is used as a variable name, each instance is assumed to correspond to a distinct variable (Clocksin and Mellish, 1981). Note that while FA specifications may contain variables, FA structures may not.

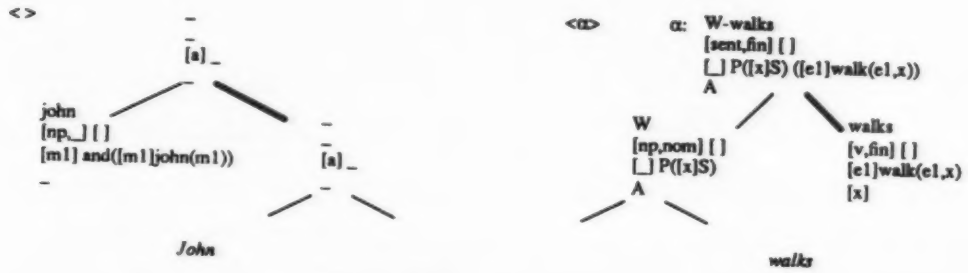

Figure 2-2: Lexical Entries.

In TUG, FA specifications are used as lexical entries. The FA specifications introduced in Figure 2-2 are the lexical entries for *John* and *walks*. Note that we have not abbreviated the InL formulae contained in these FA specifications to allow us to illustrate grammar rule application more clearly. In these FA specifications, the functor is distinguished from the argument by using darker lines for the functor branches. The actual left-to-right ordering of the different branches is not relevant. The root-sign of an FA specification contains information associated with a complex expression with the descendent nodes possessing information associated with subexpressions. Consider the lexical entry for *walks* from Figure 2-2. The phonology of the root-sign is *W-walks*, where *W* is some variable obtained from the phonology of the argument-sign. Its syntax states that it is a finite sentence, and that its slash list is empty. The partially specified InL formula contained in the semantic attribute has an unspecified index, and consists of a condition with a unspecified predicate which takes two InL formulae as its arguments

There are two types of terminal node in an FA specification. The argument node of the lexical entry for *walks* in Figure 2-2 is a *pseudoterminal* while the functor is a true terminal node. Pseudoterminal nodes are distinguished from true terminal nodes by associating *hanging edges* with the former but not with the latter. In the FA structure described by an FA specification, a pseudoterminal can correspond to either a terminal or a nonterminal node. So an FA specification may state that either a constituent has specific subconstituents (nonterminal nodes), that it has no subconstituents (terminal nodes), or that it is not known whether or not a constituent has any subconstituents (pseudoterminal nodes).

2.3. The Grammar Rule

The auxiliary list specifies subtrees of the FA specification with which other FA specifications must be unified. The single TUG grammar rule makes use of these auxiliary lists in stating how different FA specifications are related. If we adopt the notation T_{α} for an FA specification consisting of a tree T with an auxiliary list α, then the TUG grammar rule can be formulated as shown in (4).

(4) $P_{\alpha} \rightarrow P_{[A|\alpha]}, A_{[]}$

This rule states that an FA specification $A_{[]}$ consisting of a tree A (known as the auxiliary tree) with an empty auxiliary list $[\,]$ can be unified with some subtree of P (the primary tree) described by the first element of the auxiliary list $[A|\alpha]$. The result of this unification will be an FA specification that possesses a more fully instantiated version of P and an auxiliary list that consists of all but the first element from the auxiliary list associated with the primary tree.

The TUG grammar rule closely resembles the functional application rule of categorial grammar frameworks. Rules of functional application are of the form shown in (5).

(5) $\alpha \rightarrow \alpha / A,\ A$

In traditional categorial grammar, α and A correspond to categories. UCG generalizes this rule so that it applies to signs instead of categories. TUG takes this generalization one step further and essentially applies this rule to trees instead of signs. One might envisage a more general form of (4), shown in (6), which would be based on the categorial grammar rule of functional composition (7).

(6) $P_{\beta\alpha} \rightarrow P_{[A|\alpha]}, A_{\beta}$

(7) $\alpha / \beta \rightarrow \alpha / A,\ A / \beta$

The juxtaposition of the lists α and β in (6) denotes list concatenation. We shall restrict ourselves to (4), however, since it is sufficiently powerful for our needs.

The behaviour of (4) can be illustrated by examining the derivation of the sentence *John walks* from the lexical entries introduced in Figure 2-2. The tree associated with the lexical entry for *John*, like the one for *walks*, contains three signs. It specifies the constituent *John* to be an argument noun phrase with an unspecified case. This lexical entry also requires the semantic index of the root-sign of the tree to be identical to that of the functor-sign. According to the grammar rule introduced in (4), the FA specification

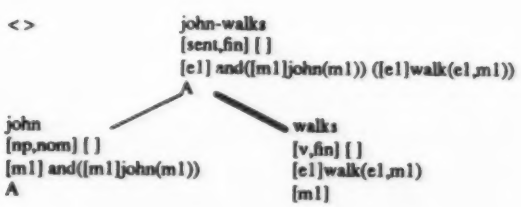

Figure 2-3: FA Specification.

for *walks* can act as the primary tree and can be combined with the FA specification for *John* (the auxiliary tree) to yield the FA specification shown in Figure 2-3. In unifying the auxiliary tree with the subtree of the primary tree labelled with α, *W* becomes instantiated to *john*, *P* to *and*, *S* to *john(m1)*, *x* to *m1*, and the semantic index of the root-sign becomes instantiated to *e1*. Note that the auxiliary list of the resulting FA specification consists of the empty list.

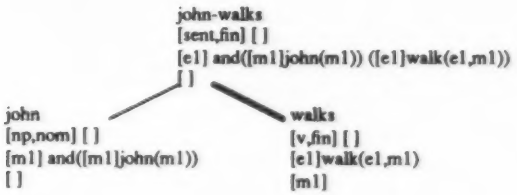

Figure 2-4: FA Structure.

An FA structure compatible with this FA specification is shown in Figure 2-4. The trees from Figures 2-3 and 2-4 differ in that the former possesses a variable *A* in the reflexive attribute of the root-sign and argument-sign, while the latter has this variable instantiated with the empty list (corresponding to a null reflexive-anaphoric context). In general, we will be interested with FA structures with null anaphoric contexts. Note that the FA specifications introduced in Figure 2-2 are also partial descriptions of this FA structure.

2.4. Lexical Templates

In proposing but a single grammar rule for our formalism, we are forced to transfer much of the information traditionally contained in grammar rules into the lexicon. Lexical templates (Shieber et al, 1983), or lexical types (Flickinger, 1987, Pollard and Sag, 1987), can be used to describe structural properties shared by a class of lexical entries, and thus provide a mechanism for capturing linguistic generalizations. Lexical entries can be defined in terms of templates, and templates themselves may be defined in terms of templates as we shall see when we introduce the templates of the lexicon in §3.1. The templates of TUG closely resemble the FA specifications that are used as lexical entries. Like lexical entries, templates contain binary trees of signs and each node in a template may have a list of templates associated with it. Unlike lexical entries, templates are not required to possess auxiliary lists. Lexical entries may be viewed as a special type of template in this regard. The three different templates introduced in Figure 2-5 describe the various relationships between the phonological information of the functor, argument, and root-signs of subtrees in FA structures. In *P-PST*, the phonology of the root-sign is that of the argument followed by that of the functor. *P-PRE* embodies the opposite relation, while *P-WRP* is used to describe the wrapping relation between phonologies (Bach, 1979, Pollard, 1984) in which the argument's phonology is embedded inside that of the functor's.

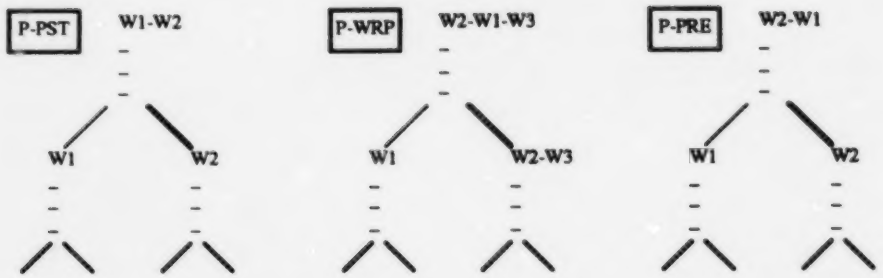

Figure 2-5: Templates for Phonological Information.

2.5. Lexical Rules

Templates are not the only means for capturing linguistic generalizations in the lexicon; lexical rules (Kaplan and Bresnan, 1982, Flickinger, 1987, Pollard and Sag, 1987) are used as well. These rules can be viewed as productions which state how to form new lexical entries from pre-existing lexical entries. Alternatively they can be seen as redundancy conditions which hold between lexical entries.

Lexical rules consist of an FA specification (called the restriction) which is separated by an arrow from another FA specification (the result). Any lexical entry that unifies with the FA specification forming the *restriction* of the rule can be modified as stipulated in the FA specification appearing as the *result* of the rule to produce a new lexical entry. Values introduced in the result replace values present in the original lexical entry. For example, we will find it useful to introduce a lexical rule shown in Figure 2-6

Figure 2-6: Lexical Rule for Null Subject Instantiation.

which, when applied to the lexical entry for any verb, will produce an alternative lexical entry corresponding to a verb requiring a nonrealized NP (ie. PRO) as a subject. This rule will be discussed in §3.4.

3. A Small Lexicon

A grammar within the TUG formalism consists of a grammar rule, a lexicon, and an FA specification that acts as a start symbol. Since we have already seen the single grammar rule, we will now introduce a representative selection of lexical entries from our grammar. At the same time, we will examine the FA

specifications obtainable from these lexical entries. We will be concentrating on constructions in which proper names, universally quantified noun phrases, indefinite noun phrases and possessive noun phrases introduce discourse markers which act as potential antecedents for reflexive pronouns.

3.1. Templates

Since FA specifications can be simplified greatly through the use of templates, we will in this section define a collection of templates that capture various relations between the information contained in a root-sign of some subtree and that of its functor and argument signs. As we have already introduced all the templates that we will need for capturing phonological relationships (Figure 2-5), we will now examine templates for syntactic, semantic and anaphoric relationships. We will then introduce some more complex templates that are defined in terms of these simple ones.

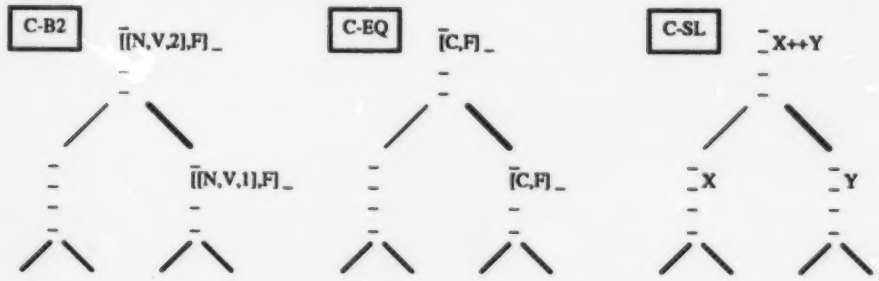

Figure 3-1: Templates for Syntactic Information.

The three syntactic templates in Figure 3-1 illustrate the different possibilities in the formation of the syntax of the root-sign from that of the functor-sign and the argument-sign. One possibility, described in the *C-B2* (category bar 2) template, has the syntactic category (which is defined in terms of the three features introduced in Figure 2-1) of the root-sign of a subtree specified to be the same as that of the functor-sign except that the *bar level* of the former is one greater than that of the latter. The second template, *C-EQ* (category equality), defines the syntactic category of the root-sign to be the same as that of the functor-sign. These two templates illustrate that the functor-sign acts as the *head* of the constituent described by the root-sign; it possesses many of the same syntactic features as the complex constituent. The *C-SL* template illustrates how the slash list of the root-sign is determined from that of the functor and argument signs. It is defined in terms of the merging operator "++" to be the result of merging the slash lists associated with the functor and argument signs. Given two lists *X* and *Y*, the result of merging the two lists (*X++Y*) is a list containing all the elements of the second list plus those of the first that are not already present in the second.

Different semantic relationships are captured by the templates introduced in Figure 3-2. These relationships are quite closely related to those discussed in categorial grammar accounts of natural language semantics (Oehrle, Bach and Wheeler, 1988). The semantic template called *S-TR* (semantic type-raising) in Figure 3-2 illustrates how the InL formula of the root-sign can be composed of the

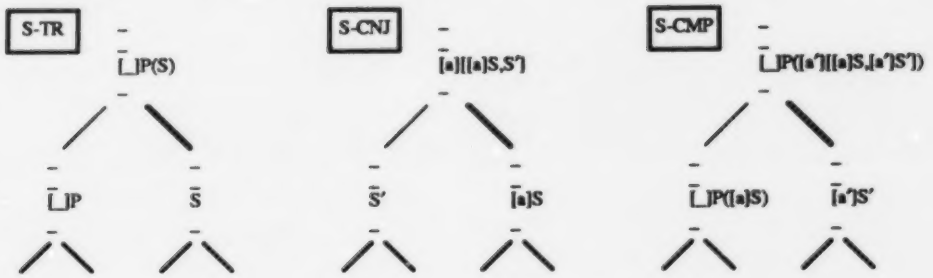

Figure 3-2: Templates for Semantic Information.

semantics of the argument-sign acting as a predicate that takes the InL formula of the functor-sign, *S*, as its argument. *S-CNJ* (semantic conjunction) is used to conjoin two semantic formulae. A conjunctive condition over the two subformulae is introduced, and the index of the complex formula is identical to that of the functor. Finally, *S-CMP* (semantic composition) describes a relation that differs from *S-TR* in that the semantic information of the functor-sign does not become the second argument of the semantic connective *P*, instead it is incorporated into the first argument. In a larger lexicon, we might need more templates to define additional relationships.

Recall that the reflexive attribute of a sign consists of a list of sorted variables corresponding to possible antecedents for reflexive pronouns. The different sorts associated with the variables mediate the agreement between a pronoun and its antecedent. For any sign in an FA structure, this members of this list are determined based on restrictions proposed by Keenan (1974) and by Pollard and Sag (1983).

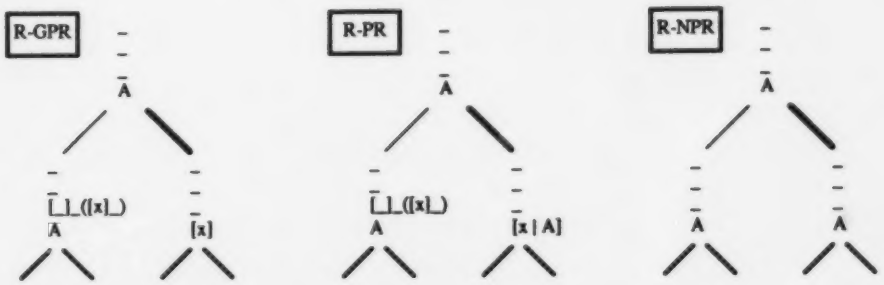

Figure 3-3: Templates for Reflexive Information.

Stated in terms of templates, these restrictions are shown in Figure 3-3.

In the template for *generalized predicative functors*, *R-GPR*, the functor's R-antecedent information consists of the the *anaphoric index* of the argument. This relationship reflects Keenan's (1974) constraint requiring an antecedent for a pronoun to be an argument of the functor containing the pronoun as discussed in (Popowich, 1989a). For a semantic formula of the form shown in (8), we define *x* to be the anaphoric index of the formula.

(8) [a]P([x]S)

In all the examples that we will be discussing here, the anaphoric index, *x*, and semantic index, *a*, of a noun phrase sign are identical. The notion of a generalized predicative functor is associated with a *semantic* locality restriction on reflexivization proposed by Pollard and Sag (1983). This restriction prevents reflexive "information" from passing through categories of a *generalized predicative* type. Conventional categories of this type include possessed nominals (like *picture of himself* in the phrase *John's picture of himself*) and verb phases. This restriction is reflected in R-GPR by having the functor-sign not contain any information from the reflexive attribute of the root-sign.

For predicative functors that are not generalized predicatives, the reflexive attribute of the functor-sign is determined by augmenting the list obtained from the reflexive attribute of the root-sign with the anaphoric index of the argument-sign. This is shown in *R-PR*. In subtrees with nonpredicative functors, the information contained in the reflexive attribute of the functor-sign is simply inherited from that of the root-sign as stated in *R-NPR*. Note that in all cases the R-antecedent information of the argument-sign is the same as that of the root-sign.

Templates and lexical entries can be defined in terms of other templates by labelling a node in an FA specification with a list of template names. The template list is displayed to the right of the node in angle brackets. For every name appearing in the list associated with a node, a copy of the corresponding template is unified with the subtree rooted at the node. For a template or a lexical entry to be well formed, this unification operation must always succeed. This is illustrated in Figure 3-4 in which a template

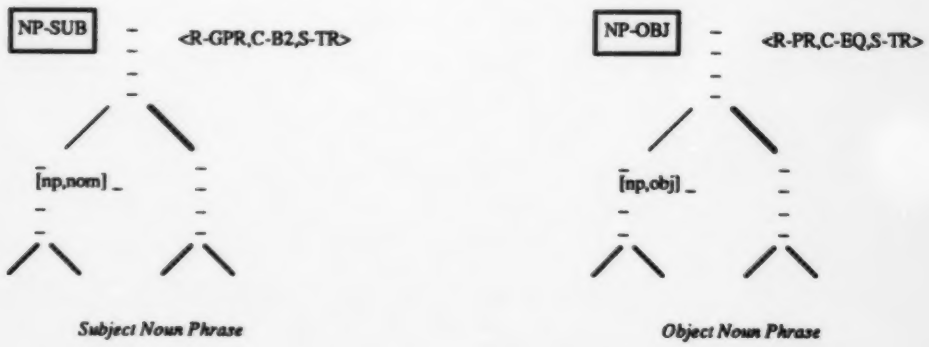

Figure 3-4: Templates for Noun Phrases.

NP-SUB describes information common to all object noun phrases. It is defined in terms of the templates *R-GPR* (since subject noun phrases are the arguments of generalized predicative functors), *C-B2* (since nonmaximal projection (ie. bar level 1) functors result in maximal projection (bar level 2) constituents when combined with a subject noun phrase), and *S-TR* (since the subject noun phrase contains the semantic connective that is to be used in determining the InL formula associated with the root-sign). A similar template, *NP-OBJ*, is defined in Figure 3-4 for describing object noun phrases. It differs from *NP-SUB* in that the functor is not a generalized predicative, and in that the root-sign and functor-signs have identical syntactic categories (as required by the presence of *C-EQ*).

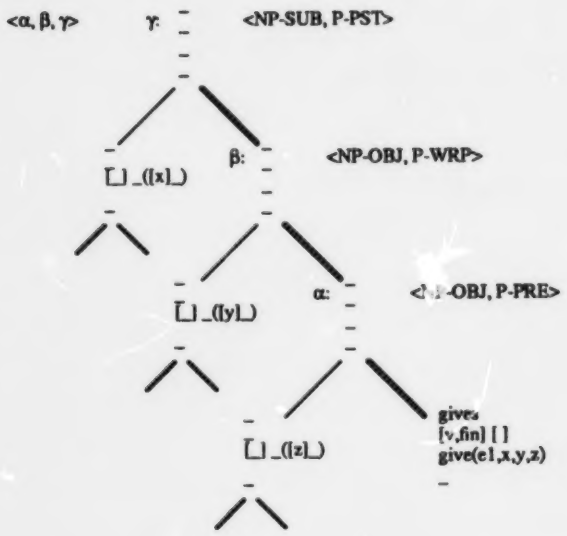

Figure 3-5: Lexical Entry for *gives*.

Finally, an example of a lexical entry defined in terms of templates is provided in Figure 3-5. This lexical entry states that the FA specification for *gives* must combine with three FA specifications in order to become a complete constituent. One of these FA specifications, α, will be an object noun phrase *(NP-OBJ)* that will be preceded *(P-PRE)* by the word *gives* in the sentence. The discourse marker associated with this noun phrase will be the final argument of the *give* predicate in an InL formula. Another of the FA specifications, β, will also be an object noun phrase, and it will appear after *give* but before the other object noun phrase in the sentence *(P-WRP)*. The remaining FA specification, γ, will be a subject noun phrase *(NP-SUB)* that will appear before the functor verb phrase in the sentence *(P-PST)*.

Each subtree of the FA specification labelled by a template name has a copy of the specified template unified with it; separate invocations of the same template do not introduce the same variables. For example, the subtree labelled α will have copies of the template *NP-OBJ* and *P-PRE* unified with it. Since *NP-OBJ* is defined in terms of *R-PR*, *C-EQ*, and *S-TR*, copies of these templates will also be unified with this subtree. This will result in a subtree containing the information shown in Figure 3-6. Note that the variable *W* is introduced by the *P-PRE* template and *A* is introduced by *R-PR*. The introduction of the variables *F* and *S* is a consequence of the restriction involving the variable *P* introduced by *S-TR*.

3.2. Reflexives and Picture-Nouns

We shall now introduce a few lexical entries required to account for the distribution of reflexives within *picture-noun* constructions as discussed by Popowich (1989a). A picture-noun construction contains a common noun (like *picture* or *book*) followed by a preposition and a noun phrase (like *of John* or *about himself*) that specify what the noun depicts or describes. After introducing these new lexical entries, we shall see how they are combined with lexical entries presented earlier in this paper forming

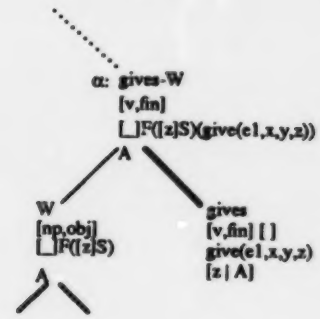

Figure 3-6: Template Expansion.

more complex FA specifications through unification.

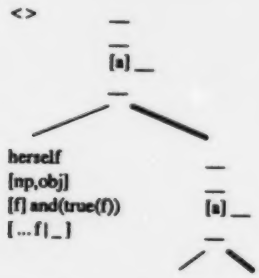

Figure 3-7: Lexical Entry for *herself*.

The lexical entry for *herself* is introduced in Figure 3-7. It contains a restriction *[... f|_]* which requires the discourse marker *f* to be a member of the reflexive attribute of the sign with phonology *herself*. Furthermore, following our convention for naming our sorted variables, its semantic index *f* is of the feminine-singular sort. The semantic attribute of the argument-sign contains the semantic connective *and* which will be used in determining the semantics of the root-sign.

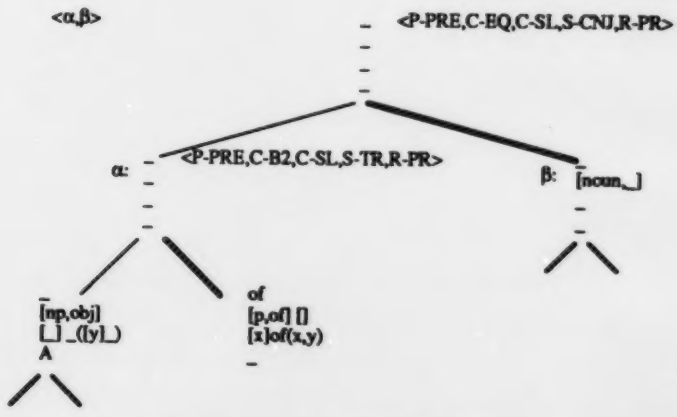

Figure 3-8: Lexical Entry for *of*.

The lexical entry for *of* as used in picture-noun constructions, is introduced in Figure 3-8. It possesses two elements on its auxiliary list, one (α) associated with an object noun phrase, and another (β) associated with the common noun modified by the prepositional phrase. The FA specification for *picture of herself*, which is shown in Figure 3-9, is obtained by unifying the FA specification for *herself* (Figure 3-7) with the subtree referenced by α, and by unifying the FA specification for *picture* with the subtree referenced by β. Lexical entries for common nouns like *picture* will consist of depth zero trees as shown in (9).

(9) <> picture
[noun,_] []
picture(n)
–

Recall that the FA specification for *herself* from Figure 3-7 possessed a restriction requiring *f* to be a member of the reflexive attribute of the sign with phonology *herself*. To represent that this restriction was

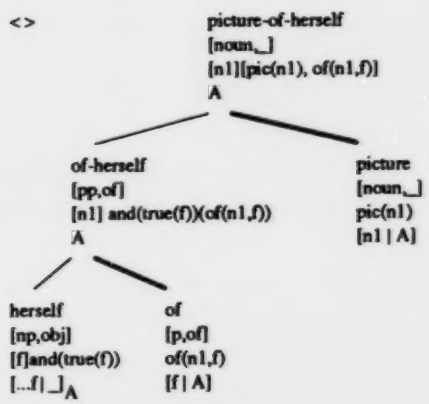

Figure 3-9: FA Specification for a Picture-Noun.

unified with the variable *A* from the lexical entry for *of*, we will subscript this restriction with *A* in the FA specification in Figure 3-9. This is just a convenient way of showing that *A* is restricted to be a list containing *f* as one of its members.

The lexical entry for the indefinite article *a* is introduced in Figure 3-10. It requires an auxiliary tree to be unified with the subtree α. Note that the tree structure specifies the noun to be a functor that takes the determiner as its argument. This is consistent with the approach adopted in HPSG (Pollard and Sag, 1987), and it allows a uniform tree structure for noun phrases with possessive and non-possessive determiners. The sign associated with the indefinite article introduces the semantic connective *and* that is used in determining the semantics of the noun phrase.

The lexical entry for *loves* introduced in Figure 3-11 contains two elements in its auxiliary list corresponding to its object and subject noun phrases. When the subtree reference by α is unified with the FA specification for *a picture of herself* (Figure 3-9), *A* from the auxiliary tree becomes instantiated to *[x]*, and *y* is instantiated to *n1* as shown in Figure 3-12. Since there is a restriction on *A* that requires *f* to

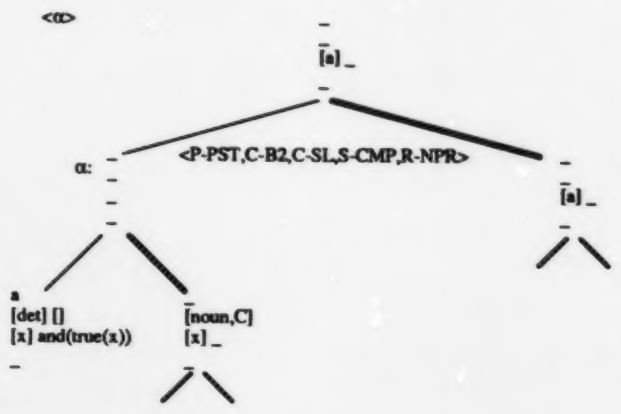

Figure 3-10: FA Specification for the Indefinite Article.

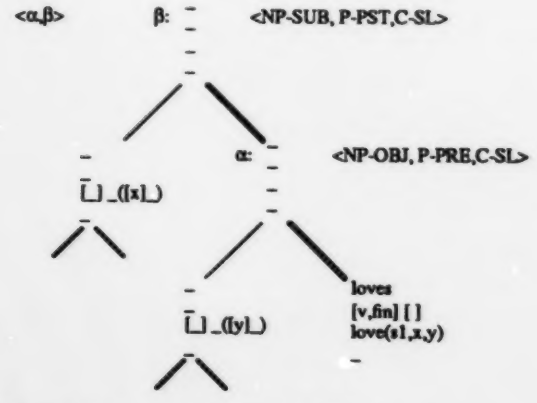

Figure 3-11: Lexical Entry for *loves*.

be an arbitrary member of the list to which *A* becomes instantiated, *f* must be unified with *x*. Note that the semantic formula *PIC(n1,f)* in Figure 3-12, which should be interpreted as stating that the entity *n1* is a picture of entity *f*, is an abbreviation for *[n1] [pic(n1), of(n1,f)]*.

When the FA specification from Figure 3-12 acts as a primary tree that is combined with an FA specification for *Mary*, *f* will be instantiated to the semantic index associated with *Mary*. The primary tree could not take an auxiliary tree possessing a semantic index of the singular-masculine sort since this would lead to a sort-conflict and unification would fail. So there would be no FA specification for a sentence like *John loves a picture of herself* unless *John* were female.

3.3. Possessives

Another class of constructions that is interesting with respect to reflexivization is possessives. Possessive constructions can be analyzed in terms of generalized predicatives in order to account for the behaviour of reflexive pronouns contained in complex noun phrases with possessive determiners.

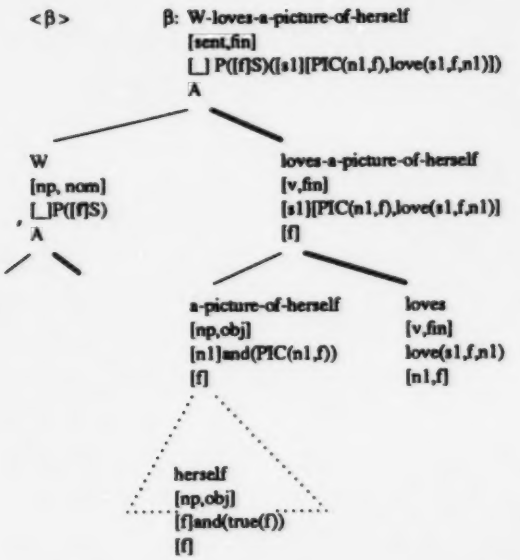

Figure 3-12: FA Specification for a Verb Phrase.

A possessive construction like *Mary's picture of herself* can be analyzed in terms of a possessed nominal (*picture of herself*) acting as a (generalized predicative) functor over a possessive noun phrase (*Mary's*) (Keenan and Faltz, 1978, Pollard and Sag, 1983). Translating this into an FA specification, a

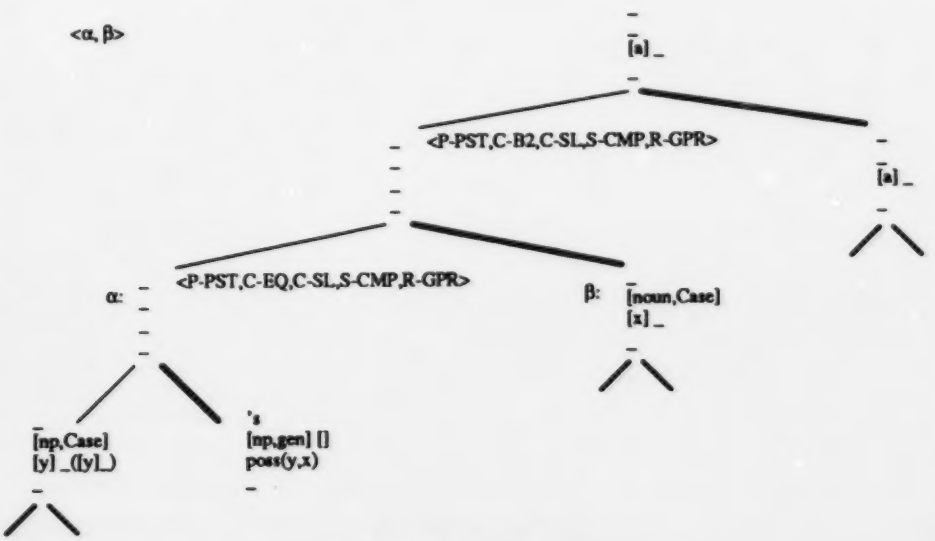

Figure 3-13: Lexical Entry for Possessive.

generalized predicative noun (*R-GPR*) takes the "possessor" as its argument, as illustrated in the lexical entry for the possessive affix *'s* in Figure 3-13. Observe that the sign associated with the possessive affix *'s* is a functor which takes a noun phrase as an argument in order to produce a genitive noun phrase.

The *Case* of this noun phrase argument should actually be restricted to be objective or nominative.[2] The constituent formed from the generalized predicative noun and its genitive argument is a noun phrase. The auxiliary list of the lexical entry contains entries for the possessed noun, β, and for the noun phrase corresponding to the possessor, α. A semantic possession condition is introduced in the sign corresponding to the possessive affix. Intuitively, the condition *poss(y,x)* means that the entity *y* possesses the entity *x*.

As mentioned in §3.1, R-antecedent information associated with the complex noun phrase is not incorporated within that of the noun since the noun is a generalized predicative. So, any markers contained in the reflexive attribute of the complex noun phrase are unavailable to reflexives contained in the subtree associated with the noun, β. Consequently, derivations like *John loves Mary's picture of himself* are blocked; there is no antecedent of the appropriate sort available to the reflexive. Since the anaphoric index of the argument-sign is introduced into the reflexive attribute of a generalized predicative functor, the marker associated with the possessor is available to reflexives contained in possessed nominals. This is illustrated the section of the FA specification for *Mary's picture of herself* included in

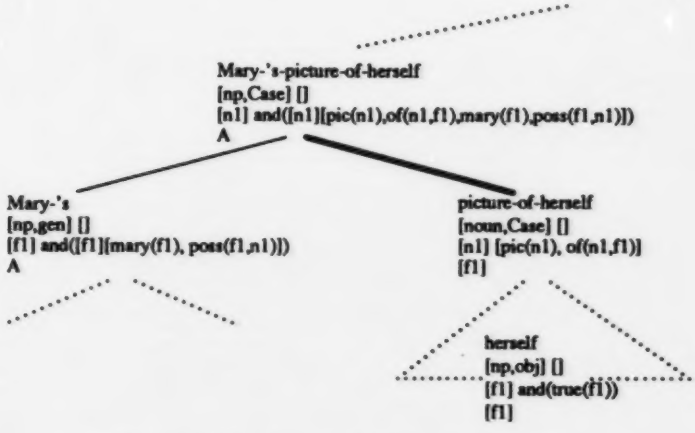

Figure 3-14: Possessive with Picture-Noun.

Figure 3-14. The index of *Mary* is contained in the reflexive attribute of *herself*, but the R-antecedent information, *A*, associated with the complex noun phrase is not included in the reflexive attribute of the generalized predicative noun. The neuter-singular-object sort anaphoric index, *n1*, of the noun phrase *Mary's picture of herself* corresponds to *picture*, not to *Mary*. Since it is this anaphoric index, not the index from the embedded noun phrase *Mary*, that serves as a possible antecedent for reflexives in more oblique arguments in a sentence, ungrammatical sentences like *John gave Mary's picture of herself to herself* are not allowed. For this sentence, the reflexive attribute of the final pronoun will contain only the anaphoric indices of *John* and of *picture*. Based on the same principles, sentences like *Jane told Mary's brother about himself* will be grammatical.

---

[2]This could be done by arranging the various cases in a subsumption hierarchy and introducing complex cases. For example, unification of the complex case *nom+obj* with *nom* would result in *nom*, while unification with *gen* would fail.

3.4. Controlled Complements

Controlled complement constructions are interesting with respect to reflexivization since locally bound reflexives in such complements can have antecedents which are from outside the complement clause. In this section we will examine infinitival complements.

Consider the following examples in which the infinitival complements are bracketed.

(10) John wants [to walk].

(11) John persuaded himself [to love Mary].

Each of the bracketed clauses is lacking a subject noun phrase. The missing subject is usually considered to be an empty pronominal noun phrase called *PRO* (Chomsky, 1973) which is dependent on some other noun phrase just like a pronoun is dependent on its antecedent. This dependency involving PRO, which is called *control*, is associated with *equi* verbs like those introduced in (10)-(11). The PROs that are implicit in (10)-(11) are shown explicitly in (12)-(13).

(12) *John* wants [*PRO* to walk].

(13) John persuaded *himself* [*PRO* to love Mary].

The antecedent of PRO, which is called its *controller*, is italicized in these examples.

The use of FA specifications as lexical entries allows local relationships like control to be stated explicitly in a lexical entry. Just as the reflexive anaphoric relationship is captured by unifying the anaphoric indices of the reflexive and its antecedent, control is expressed by the unification of the anaphoric indices of PRO and its controller. This requires the presence of a sign for PRO, but it does not mean that a separate lexical entry for PRO is required. The sign corresponding to PRO can be incorporated into the lexical entry for the equi verb. In this lexical entry, both the sign for PRO and the one for its controller will have the same index.

Figure 3-15 introduces the lexical entry for *persuades* which is responsible for *object-equi* constructions (ie. constructions in which the object is the controller) as exemplified by sentence (14).

(14) John persuades Mary to wash herself.

The subtree α of the FA specification for *persuades* corresponds to an infinitival "sentence." This clause is an argument of *persuades*. The argument-sign of α is required to be a noun phrase with a null phonology, ε, with a semantic attribute structured like that of a pronoun. It corresponds to *PRO*, whose index *y* is obtained from that of its controller, the object of the main verb.

Although we have examined the FA specifications of verbs that take infinitival complements, we have not yet considered the structure of the infinitival complements themselves. Syntactically, infinitival complements appear to be verb phrases. In our grammar, the FA specification for a verb phrase possesses a *non-empty* auxiliary list. This list contains a description of the subtree with which the FA specification for the subject noun phrase is to be unified. But if an equi verb is to take the FA

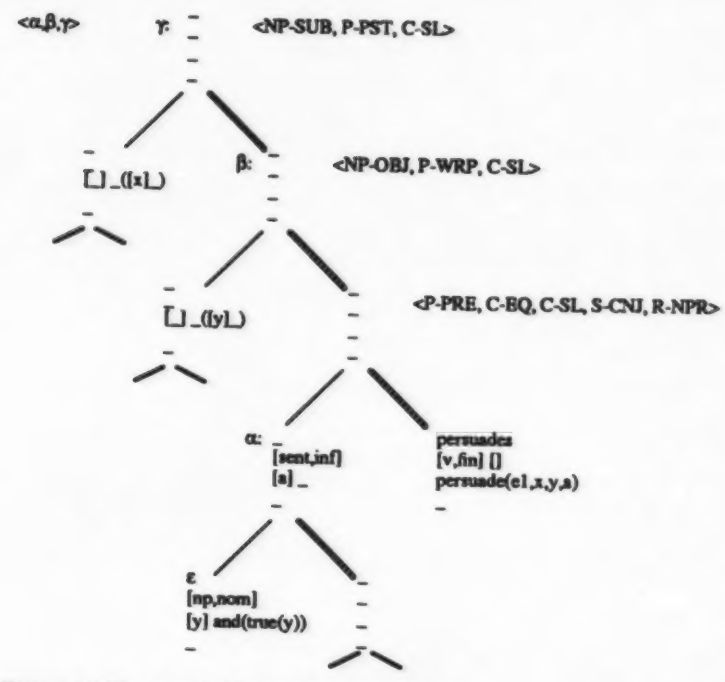

Figure 3-15: Lexical Entry for *persuades*.

specification for a verb phrase as an auxiliary tree, the verb phrase must have an *empty* auxiliary list — the TUG grammar rule requires an auxiliary tree to have an empty auxiliary list. To allow FA specifications associated with verb phrases (and constructions like adjectival phrases) to act as auxiliary trees, their auxiliary lists must be emptied and information that is normally supplied by their own auxiliary trees must be specified. One could introduce an additional grammar rule which would allow FA specifications with non-empty auxiliary lists to act as auxiliary trees. In keeping with the highly lexical approach adopted in TUG, we will instead propose a lexical rule.

For example, the lexical entry for a transitive verb like *loves* consists of an FA specification containing two elements in its auxiliary list, one describing where the auxiliary tree for the subject noun phrase should be unified and the other the location for the object noun phrase. The lexical rule for null subject instantiation (Figure 2-6) states that there is an additional lexical entry for *love* containing only the element corresponding to the object noun phrase. This alternative lexical entry, which is shown in Figure 3-16, possesses an argument-sign describing a noun phrase with a null phonology and an essentially vacuous semantics (the semantic attribute contains the condition *true* which is always satisfiable). After this lexical entry has been combined with an auxiliary tree corresponding to an object noun phrase, it will possess an empty auxiliary list. In this way we will have an FA specification for a verb phrase that can be used as an auxiliary tree to be combined with primary trees requiring verb phrase complements. It turns out that the use of this alternative lexical entry is not restricted to controlled complement constructions. It can also be used with adverbials that modify verb phrases, and in constructions involving coordination of verb phrases.

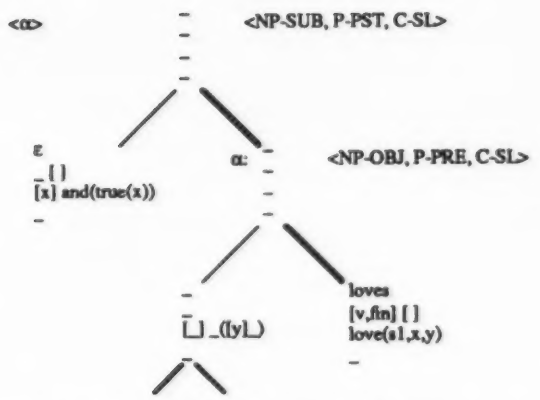

Figure 3-16: Lexical Entry for Verb in Controlled Complement.

One may wonder why we do not instead propose a lexical rule in which the argument-sign associated with the subject is simply missing instead of having a subject sign possessing a null phonology and a vacuous semantics. That is, why not propose a lexical entry for *loves* consisting of only the subtree α of Figure 3-16. The reason for not doing this is that the null noun phrase sign that corresponds to PRO and the information contained in this sign is used to mediate the control relationship. Without this sign, it would be very difficult for verbs like *persuades* to state that the discourse marker associated with the controller is the same as the discourse marker corresponding to the subject of the subordinate verb phrase. Furthermore, the presence of this sign is essential to our account of reflexivization.

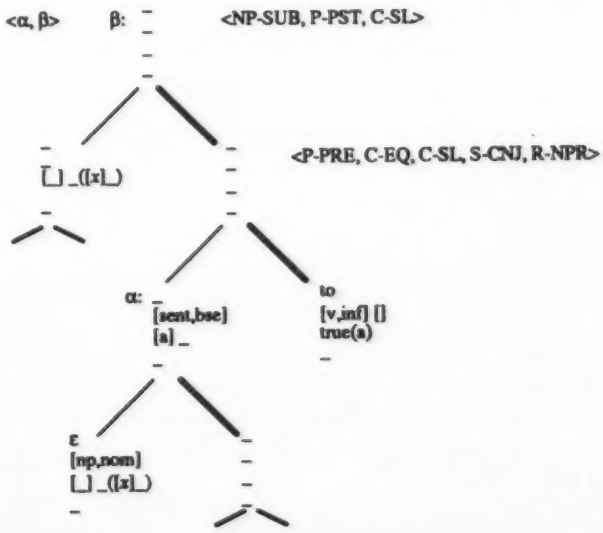

Figure 3-17: Lexical Entry for Auxiliary Verb *to*.

The lexical entry for *to* as used in infinitival complements is introduced in Figure 3-17. Adopting the treatment proposed in Generalized Phrase Structure Grammar (GPSG) (Gazdar et al, 1985, p.114) and in

HPSG (Pollard and Sag, 1987, p.61), *to* is treated as a "defective auxiliary verb." Its auxiliary list specifies two auxiliary trees, one corresponding to a base form verb phrase α and another for the subject β of the infinitival complement. Observe that the index of the null subject of the base form complement is unified with that of the subject of the infinitival complement (as highlighted with the italic *x* in Figure 3-17). The FA specification acting as the auxiliary tree which is unified with α must have a null subject (as specified by the information contained in the argument-sign of α) and must have an empty auxiliary list (as imposed by the TUG grammar rule). So FA specifications of sentences with non-null subjects cannot be used as auxiliary trees here.

In the FA specification for a sentence like (14), the index of *Mary* appears both as an argument of the semantic predicate *persuade* and as an argument of the semantic predicate *wash*. As a result of the treatment of control, due to the structure of the lexical entry for *to* and due to the general principles concerning the distribution of R-antecedent information, the discourse marker associated with *Mary* is contained in the the reflexive attribute of the reflexive pronoun even though *Mary* is not within the same clause as the reflexive.

An analysis for an ungrammatical sentence like the one introduced in (15) is not allowed; the discourse marker corresponding to *John* is not available to the reflexive pronoun.

(15) *John persuades Mary to wash himself

Only the anaphoric index of the null subject of the verb *wash* is available to the reflexive, and this index is unified with that of the controller *Mary*.

A verb like *promises* which takes a complement that is controlled by the subject will have a lexical entry almost identical to the one for *persuades*, except that the index of the null noun phrase is specified to be identical to the anaphoric index of the subject instead of that of the object. Since this distinction between object and subject control is highly dependent on the lexical entry, it seems appropriate to specify the various control types explicitly in the various lexical entries. Due to the presence of subject control, sentences like *John promises Mary to wash himself* would be allowed, while *John promises Mary to wash herself* would not be. In subject-equi sentences like these, only the marker associated with *John* is available to an embedded reflexive.

3.5. Unbounded Dependencies

An unbounded dependency is a relationship between two syntactic components of a sentence in which the related constituents are not required to be within some bounded distance from each other. Often, unbounded dependencies are described in terms of "movement" whereby one constituent is "moved" some unbounded distance away from its canonical location and a binding relation is established between the moved constituent and a *trace* inserted in the canonical location (Chomsky, 1975). We will restrict our discussion to the distribution of reflexive pronouns in unbounded dependency constructions associated with relative clauses.

Although there are several classes of relative clauses, the discussion here will be limited to *restrictive* relatives. Restrictive relative clauses can be distinguished from *appositive* relatives by noting that the former generally restrict a head noun, while the latter give additional information about a noun phrase. These relatives also differ in the type of relative pronoun that may be used; appositive relatives cannot contain the relative pronoun *that*. Often, these two classes of relatives can also be distinguished in text by the presence of commas around the clause. The sentence introduced in (16) contains a restrictive relative while (17) contains an appositive relative.

(16) A man that loves a woman loves himself.
(17) John, who loves Mary, loves himself.

Relative clauses contain a relative pronoun plus a sentential component which is "missing" a noun phrase. In some cases the relative pronoun can be omitted, but we will not be concerned with the details of this here (see Popowich, 1988). The location of the missing noun phrase is often referenced by the insertion of a trace *t* into the sentence. A binding relation is established between the relative pronoun and the trace as illustrated by the subscripts in (18).

(18) a woman that$_i$ John wants his brother's son to marry t_i

Our TUG based proposal for relative clauses relies on the use of the slash list for establishing unbounded dependencies. Recall that the slash list forms part of the syntax attribute of a TUG sign. Its value can either be an empty list *nil* (ie. [ ]) or a list of signs which mediate an unbounded dependencies. Since a sign contains more than just syntactic information, an unbounded dependency can be more than just a syntactic relationship in TUG. For example, the sign associated with a clause containing a noun phrase trace would be of the form shown in (19) where the slash list is shown in bold.

(19) John-loves-ε
[sent, fin] , [ε; [np,ob]; [x]and(true(x)); _]
[s1] [john(m1), love(s1,m1,x)]
–

We will adopt the working hypothesis that the TUG slash list may contain at most one element. This is in accordance with the treatment of slash information contained in the *SLASH* features of GPSG and HPSG. Under this assumption and assuming that template *C-SL* holds at every nonterminal node in an FA structure, if a functor-sign and an argument-sign in some FA structure both have a non-nil slash list, then these values must be the same. If they did not — the slash list of the functor-sign were *[X]* and that of the argument-sign were *Y* (where *X* and *Y* were different) — then the slash list of the root-sign (which would be the result of merging these two lists) would contain both *X* and *Y*. This violates our assumption that the slash list may contain at most one element. The different possible relationships between the slash lists of a root-sign and that of its functor and arguments signs is summarized in (20).

(20) *functor-sign*	*argument-sign*	*root-sign*
[ ]	[ ]	[ ]
[ ]	[X]	[X]
[X]	[ ]	[X]
[X]	[X]	[X]

In most instances, one of the first three of these cases will hold. The fourth case is relevant for constructions involving parasitic gaps (Engdahl, 1983), which will not be discussed here (see Popowich, 1988).

For expository purposes, we will be discussing unbounded dependencies in a somewhat procedural manner where the trace introduces an unbounded dependency that must be resolved by the relative pronoun. Our treatment of unbounded dependencies can be broken down into three stages (cf. Gazdar et al, 1985, § 7.1). First, during the *introduction* of the unbounded dependency a sign containing information relevant to the dependency is introduced into the slash list of the trace. The *propagation* stage entails passing this information up through the tree. Finally, the *resolution* of the unbounded dependency involves the removal of the information from the slash list.

Our restrictions on slash lists also turn out to be very similar to the GPSG treatment of the *SLASH* feature, assuming the functor in an FA specification is treated as a *head*. Since SLASH is a head feature in GPSG, the head feature principle requires it to be the same on the mother and head daughter constituents (ie. root-sign and functor-sign). It is not inherited by lexical heads (ie. functor-signs associated with terminal nodes in an FA specification). GPSG possesses an additional principle (the foot feature principle) which prevents two daughters from having different elements contained in their respective SLASH features (ie. the signs contained in the slash lists of a functor and argument sign must be the same). The parasitic relationship exhibited by SLASH features is a consequence of the interaction of the foot and head feature principles — if a non-head daughter possess a SLASH feature then so must its mother (foot feature principle) and so must the head daughter (head feature convention).

In an FA specification, the sign associated with a relative pronoun will have category *np*, and its semantic attribute will be like that of other pronouns. The category of the sign associated with the relative clause will be *sent*. If for the moment we ignore details concerning R-antecedent information, we can tentatively introduce the lexical entry for the restrictive relative pronoun *that* as shown in Figure 3-18.

The lexical entry for the relative pronoun removes a sign from the slash list of the root-sign of subtree α which contains information responsible for mediating an unbounded dependency. The slash list associated with the complete relative clause (ie. the argument-sign of the FA specification in Figure 3-18) is stipulated to be empty in the lexical entry. Since the syntactic and semantic attributes of the sign in the slash list is specified to be the same as those associated with the relative pronoun, a binding relationship will be established between the relative pronoun and the trace that introduces the information present in the slash list. Since the relative clause gives more semantic information about the entity described by the head noun, the index of the noun, *x*, and the index of the relative pronoun are the same. The appearance

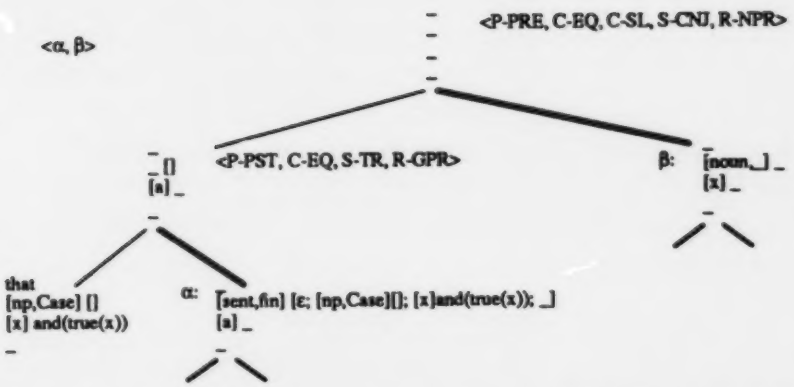

Figure 3-18: Lexical Entry for Restrictive Relative Pronoun (preliminary formulation).

of ε in the phonology of the sign contained in the slash list reflects the fact that there is a noun phrase with null phonology in the relative clause (ie. a trace). For the root-node of the FA specification, its slash information is related to that of its functor and argument signs according to the relationship stated in template *C-SL*.

There are several ways in which the introduction (as opposed to the resolution) of an unbounded dependency can be incorporated into TUG. In GPSG, there are special grammar rules for both introducing and resolving unbounded dependencies. We have so far been able to avoid the introduction of extra grammar rules and there is no need to resort to such an alternative for these cases either.

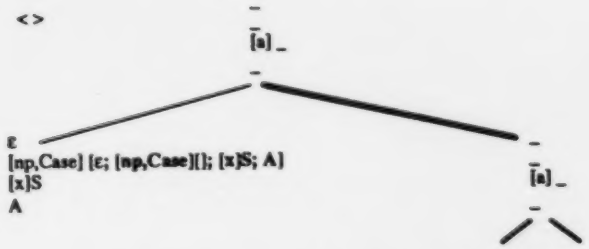

Figure 3-19: Lexical Entry for a Trace.

Instead of introducing a new rule, a separate lexical entry for the trace can be proposed as shown in Figure 3-19. In this lexical entry, the sign associated with the trace possesses a slash list which contains all the phonological, syntactic, semantic and R-antecedent information of the trace. Aside from the presence of a non-empty slash list, its structure closely resembles that of a pronoun or proper noun.

Let us now turn our attention to the distribution of reflexives in relative clauses. Consider the following pair of expressions.

(21) a man that$_i$ (Mary told t_i about himself$_i$)

(22) *a man that$_i$ (Mary told himself$_i$ about t_i)

In the FA specification for (21), it is the anaphoric index of the trace that is chosen as the index of the reflexive pronoun — the trace acts as the antecedent for the reflexive. The anaphoric index of the trace is in turn obtained from the information contained in the slash list of the relative clause. This information is introduced by the relative pronoun which stipulated this index to be the same as that of the head noun *man*. For the expression introduced in (22), the reflexive attribute of the reflexive pronoun will contain only the discourse marker associated with *Mary*. The marker associated with the relative pronoun is not contained in this attribute due to the presence of the generalized predicative associated with the expression *told himself about*. The anaphoric index of the trace is not present in this attribute due to its location in the sentence: the trace is a more oblique argument of the verb than the reflexive. Since the reflexive attribute of the masculine reflexive pronoun does not contain a marker of the masculine sort, anaphora resolution fails. So, like in controlled complements in which the anaphoric index of the null noun phrase is used to introduce the marker of the controller as a potential antecedent, the anaphoric index of the trace is used to introduce into the relative clause the marker associated with the head of a relative clause construction.

The most interesting constructions involving relative clauses and reflexives are those that contain a reflexive in the head of the nominal. This is illustrated in (23).

(23) Mary wrote a story about herself that John liked.

Generally, the anaphoric properties of such reflexives seem to be the same as those that are contained in similar constructions without relative clauses — they find their antecedents in the main clause, not the relative clause. This is illustrated by the following ungrammatical sentence.

(24) *Mary wrote a story about himself that John liked.

However, there are exceptions to this behaviour.

(25) A picture of himself that John loves was painted by Mary.

When the noun phrase containing the relative is the topic of the sentence, as in (25), it appears that an anaphoric relationship with an antecedent from the subordinate clause is possible. Furthermore, there are some sentences in which a reflexive contained in the head noun can take an antecedent from either the subordinate (downstairs) clause or the main (upstairs) clause. Consider the following sentence which is based on examples from (Kuno, 1987,6.20,6.22).

(26) Mary threw away a picture of herself that Susan took in Maine a few years back.

Either *Mary* or *Susan* can act as the antecedent for the reflexive.

As Jackendoff (1972) notes, the grammaticality of constructions involving reflexives in the head nouns of relative clause constructions is highly dependent on the choice of verbs appearing in the main and relative clauses. Kuno (1987) introduces a series of principles for determining the relative grammaticality of sentences containing reflexives. He notes that factors such as focus and perspective can affect acceptability. Since we are interested in determining the possible antecedents for reflexive

pronouns, and not in the relative acceptability of the possible antecedents, we will be adopting a "least restrictive" approach to reflexives in relative clauses. We will determine the maximal set of possible antecedents. Other factors, which are not of concern to us here, can then be responsible for determining the degree of acceptability for members of this set.

In the most general case, the R-antecedent information available to a reflexive contained in the head noun of a relative clause can come from both the upstairs and downstairs clause. Based on this

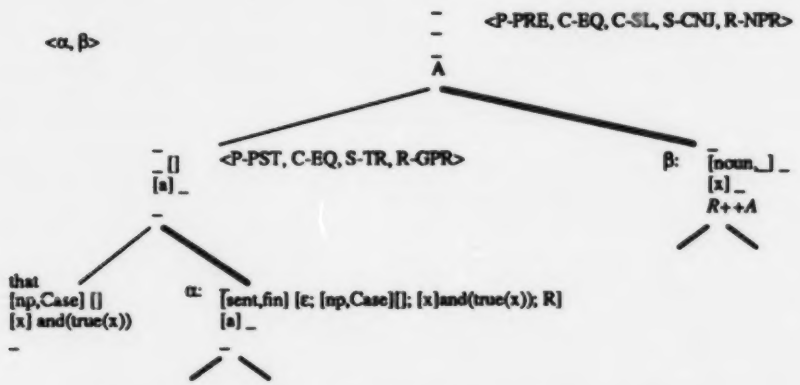

Figure 3-20: Lexical Entry for Restrictive Relative Pronoun.

observation, we will propose a lexical entry for the relative pronoun shown in Figure 3-20. The reflexive attribute of the head noun consists of the R-antecedent information *A* of the root-sign (upstairs clause) merged with the R-antecedent information *R* from the slash list of the relative clause (downstairs clause). Again, this merging is described by the expression *R++A*. With respect to anaphora, the head noun can act as if it were located where the trace is since it contains all the R-antecedent information associated with the trace. This anaphoric dependency of the head noun on the trace is mediated through the slash list.

Figure 3-21 contains a section of an FA specification associated with a simplified version of (26), namely

(27) Mary loves a picture of herself that Susan took.

Observe that the R-antecedent information of the head noun functor-sign is obtained from the slash list of the subordinate sentence (displayed in bold print) and from the reflexive attribute of the root-sign (displayed in italic). This results in the markers *f1* and *f2*, corresponding to *Susan* and *Mary* respectively, both being available to the reflexive pronoun contained in the head noun. The analysis presented in Figure 3-21 is for the case where *Susan* acts as the antecedent. Since the downstairs R-antecedent information comes from the trace, arguments in the downstairs clause that are more "oblique" than the trace cannot serve as antecedents for a reflexive in the head noun. This prevents FA specifications for ungrammatical sentences like the following.

(28) *Mary loves the picture of himself that$_i$ (Susan gave t_i to John.)

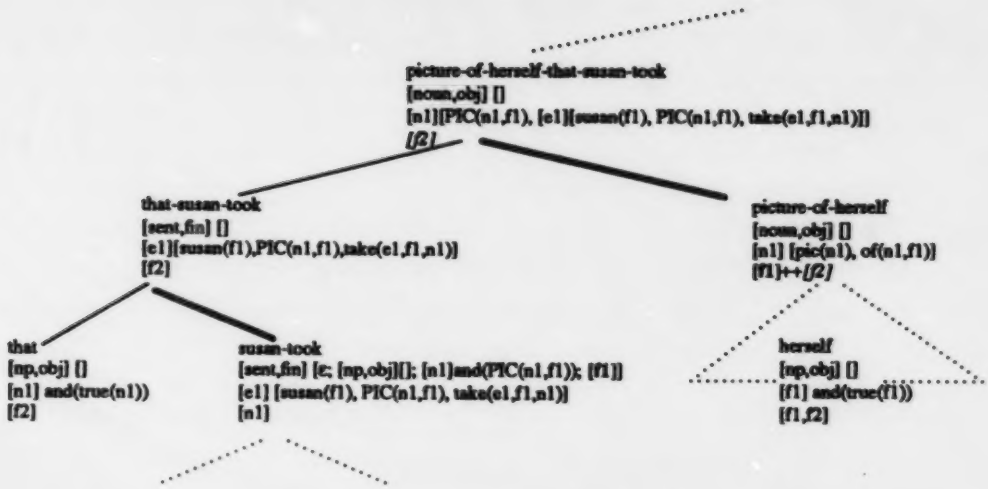

Figure 3-21: Reflexive in Relative Head.

We have outlined an account for the unbounded dependencies appearing in relative clauses which relies on information contained in the slash list to mediate the dependency. The same approach can be applied to topicalization and to cleft constructions (Popowich, 1988). As the slash list contains the phonological, syntactic, semantic and R-antecedent information associated with the trace, unbounded dependencies are treated as more than just syntactic relationships between a constituent and its trace. Through the use of the slash list, the semantic information associated with a trace can be obtained from the constituent involved in the unbounded dependency. This means that the constituent can behave semantically as if it were located in the position occupied by its trace. The R-antecedent information associated with the constituent can access the R-antecedent information associated with the trace plus information that is locally available. Consequently, unbounded dependency constituents can obtain their R-antecedent information from more than one source — the R-antecedent information from the trace is merged with the R-antecedent information that is normally obtained from the environment of the unbounded dependency constituent. They have the potential to choose an antecedent either from the clause in which they are contained or from the clause in which their trace is contained.

4. A Parser

Having outlined the various kinds of lexical entries that we include in our grammar for a small fragment of English, we can now consider how to process such a grammar in a natural language processor. There are established parsing techniques for context-free grammars (e.g., shift-reduce parsing and chart parsing) that have been adapted to handle unification-based grammar formalisms (Shieber, 1985, Proudian and Pollard, 1985, Calder, Moens, and Zeevat, 1986, Seiffert, 1987). We will consider how the active chart parsing paradigm (Kay, 1973, Kaplan, 1973, Thompson, 1981) can be applied to TUG.

4.1. Chart Parsing

Adopting the terminology used by Thompson (1981), a chart consists of edges which span segments of the input sentence. These edges contain information about different possible analyses of input segments. Each edge has a starting position, ending position, category description, list of expectations and list of constituents. Edges with no expectations are called *inactive*, while the others are called *active*. The first stage of parsing involves introducing an inactive edge into the chart for each word in the sentence. This edge is assigned the lexical category of the word. The *fundamental rule* of chart parsing states that if an active edge *A* meets an inactive edge *I* and the category of *I* satifies the current expectation of *A* then a new edge is built. The starting position of this new edge is that of *A*, its ending position is that of *I*, its category is the category of *A*, its set of constituents contains the edge *I* plus all the consituents of *A*, and its expectations are those of *A* except that the expectation corresponding to the edge I is not included. Neither *A* nor *I* is modified by the fundamental rule.

The fundamental rule applies whether a top-down or a bottom-up strategy is used. These two strategies differ in how the grammar rules interact with the construction of the chart. The grammar rules are used to create active edges containing expectations. These edges can then be combined with other edges according to the fundamental rule. The bottom up strategy requires that whenever an inactive edge is added to the chart, for all the rules in the grammar whose right hand side begins with this edge's category, an empty active edge (ie. an edge with no constituents) be added at the starting position of the inactive edge. The category of the new edge will be obtained from the left hand side of the grammar rule. The expectations will be obtained from the right hand side of the grammar rule. Chart parsers may also contain a *redundancy check* which is required for left-recursive grammars to ensure that the same edge is not added to the chart more than once. Since the single grammar rule of TUG is not left-recursive, there is no need for the redundancy check.

The various concepts associated with chart parsing are best illustrated with a simple context-free grammar like the one introduced in (29).

(29) (a) s → np, vp
(b) np → *John*
(c) vp → *walks*

Figure 4-1: Initial Chart.

To parse the sentence *John walks* according to this grammar, we first introduce inactive edges associated with the words of the sentence as shown in Figure 4-1. The edges are labelled with unique names (*i1*, *i2*), the categories of the nonterminal symbols that introduce them (*np*, *vp*), lists of expectations (which are empty, [ ]), and lists of constituent edges (which are also empty). Following the

bottom-up processing stategy, the introduction of edge *i1* to the chart will also result in the introduction of the active edge *a1*, since the category of *i1* (*np*) is the leftmost symbol on the right hand side of grammar rule (a). The active edge *a1* will have category *s* and will have expectations (*[np,vp]*) corresponding to the

Figure 4-2: Rule Invocation.

symbols from the right hand side of (a) as illustrated in Figure 4-2. Note that the active edge *a1* in Figure 4-2 meets the inactive edge *i1*; *a1*'s ending position is the same as *i1*'s starting position. According to the fundamental rule, since the category of *i1* matches the current expectation of *a1*, this results in the introduction of a new active edge *a2* to the chart. This new active edge (*a2*) which is shown in Figure 4-3

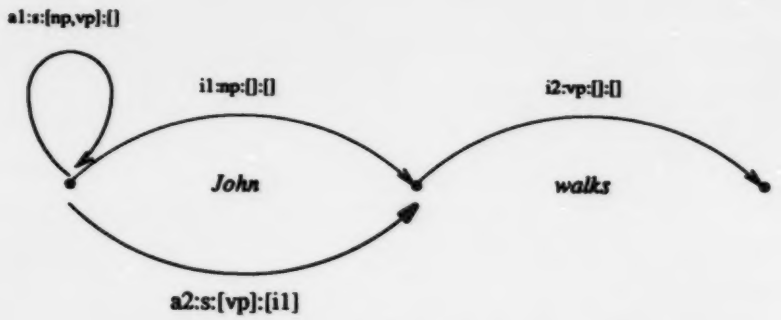

Figure 4-3: Initial Application of Fundamental Rule.

will have *i1* as one of its constituents, and will still have the *vp* expectation which it obtained from *a1*. The category of *i2* (*vp*) matches the remaining expectation of *a2* to produce a new edge which has no expectations, and which has *i1* and *i2* as constituents. This new edge which is introduced in Figure 4-4 has no expectations and is thus an inactive edge. Since it is an inactive edge that spans the entire sentence, it corresponds to a successful parse. Its constituents list can be used to construct the actual parse tree.

4.2. A TUG Chart

In adapting the chart parser to work for TUG, we need to allow an edge to contain an FA specification instead of a simple category name. Also, it will need to contain a list of constraints that are associated with a specific FA specification like those introduced by reflexive pronouns which require a discourse marker to be an arbitrary element of a list. Before we discuss these constraints, we must first consider how to represent a tree structure within the chart.

Given that our chart parser is written in Prolog, FA Specifications are represented as terms. Note

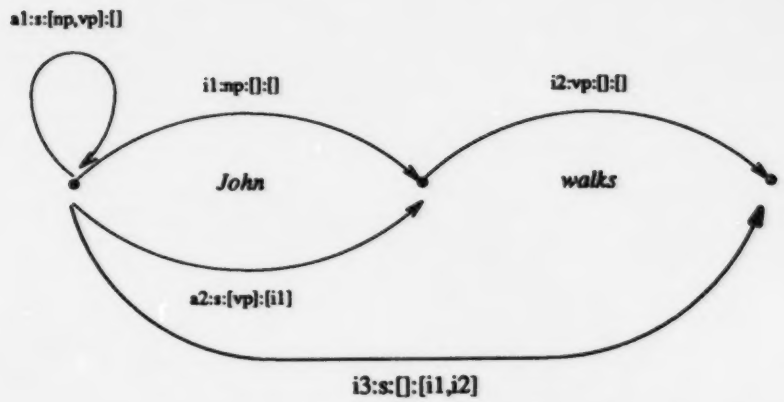

Figure 4-4: Final Application of Fundamental Rule.

that if we were to provide a LISP implementation we would probably want to use lists instead. Binary trees are encoded as terms of three arguments. The first argument corresponds to the root-sign of the tree, with the second and third arguments terms associated with the functor and argument subtrees, respectively. Trees of depth one are encoded as terms having the atom *nil* placed in the functor and argument subtree positions. Trees with hanging edges will have variables appearing in these positions. Signs are represented as terms with four arguments corresponding to phonology, syntax, semantics, and R-antecedent information. InL formulae are represented as terms with three arguments corresponding to the index, predicate name, and list of arguments. Indices themselves are treated as terms with two arguments, the first of which contains information about its sort (either entity (x(G)), state (s) or event (e)), and the second of which is a unique identifier used to distinguish different variables. The different entity sorts are distinguished by different values for *G*, so *x(m)* is used for masculine entities, *x(f)* for feminine and *x(n)* for neuter. These different encodings of information are illustrated in the term representation of

```
tree( sign(_, _, inl(Index,_,_), _),
      tree( sign(_, _, inl(Index,_,_), _),
            _,
            _)
      tree( sign(herself,
                 syn(cat(+n,-v,2),obj,_),
                 inl( i(x(f),I), and, [inl(i(x(f),I),true,[i(x(f),I)])]),
                 A),
            nil,
            nil))
```

Figure 4-5: Term Representation of a Tree.

the lexical entry for *herself* in Figure 4-5. The actual FA specification for *herself* was introduced in Figure 3-7 and is repeated here as Figure 4-6.

The auxiliary list of an FA specification is a list of subtrees with which other trees must be unified. There are two simple ways in which the auxiliary list of an FA specification can be represented: either explicitly or implicitly. The explicit approach has the auxiliary list containing a list of terms each of which corresponds to some subtree, while the implicit approach involves the auxiliary list containing a list of *binary path specifications* each of which describes how to access some subtree. In our description of the parsing system we will adopt the implicit approach since it greatly simplifies the displaying of terms.

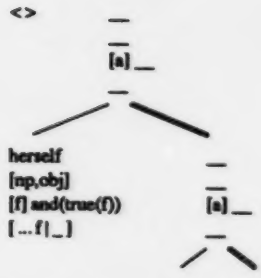

Figure 4-6: Lexical Entry for *herself*.

A binary path specification simply consists of a list of the symbols *f* and *a*. Starting with the root of a tree, the subtree described by a binary path specification is the one accessed by traversing a functor branch for each *f* and an argument branch for each *a*. This is equivalent to treating the tree as a finite state automaton whose functor branches are labelled with *f* and whose argument branches are labelled with *a*. The root node is the start state, and the binary path specification is the input to the automaton. This means that the binary path specification *[f,a]* will refer to the argument of the functor of the tree and the empty path *[ ]* will refer to the entire tree.

```
tree(sign(W-walks,
          syn(cat(-n,+v,2),fin,SL),
          inl( _, P, [inl(i(x(G),I),S,Args), inl(i(e,1),walk,[i(e,1),i(x(G),I)] )]),
          A),
     tree(sign( walks,
                syn(cat(-n,+v,1),fin,SL),
                inl( i(e,1), walk, [i(e,1),i(x(G),I)] ),
                [i(x(G),I)]),
          nil,
          nil)
     tree(sign(W,
                syn(cat(+n,-v,2),nom,SL),
                inl( _, P, [inl(i(x(G),I),S,Args)]),
                A),
          _,
          _))
/ [ [] ]
```

Figure 4-7: Term Representation of an FA Specification.

The internal representation for the FA specification associated with *walks* is shown in Figure 4-7. The tree and the auxiliary list are separated by a slash in the term notation. This is equivalent to treating an FA specification as a term with two arguments, the first corresponding to the tree and the second to the auxiliary list. Observe that the auxiliary list contains a single binary path specification ([ ]) which will reference the root of the tree. We will defer our discussion of the treatment of the slash list *SL* until §4.3.3.

Edges are represented as terms with seven arguments. The different arguments of an *edge*-term contain an FA specification, a list of pending constraints associated with the FA specification, a list of expectations (empty for inactive edges), a list of the edge numbers associated with the edges which were used to build the current edge, the starting position of the edge in the input list, the ending position, and an edge number. For example, the edge associated with the word *herself* in a sentence like *Mary loves herself* would be structured as shown in (30).

(30) edge(α / [], [member(i(x(f),I), A)], [], [], [herself], [], 1)

To simply the discussion, α in (30) stands for the term introduced in Figure 4-5. This edge contains the FA specification for *herself* (α / []), along with a constraint, [member(i(x(f),I),A)], requiring the index of the pronoun to be an element of the reflexive attribute. This edge has no expectations and is thus an inactive edge. It also has no constituents. The starting and ending position of the edge ([herself] and [ ], respectively) are represented by lists of words appearing to the right of the position, as is usually done to represent sentence positions in Prolog Definite Clause Grammar parsers (Clocksin and Mellish, 1981). The final argument position of (30) contains the edge number, 1.

4.3. Chart Parsing TUG in Prolog

The FA specifications corresponding to different lexical entries are stored as unit clauses in the Prolog database. For lexical entries defined in terms of templates, these templates are all expanded before the lexical entry is stored. Thus, template definitions do not need to be processed by the chart parser itself. They are used only in the description of the lexicon. Similarly, lexical rules are used only for describing the lexicon and are not required by the chart parser. They are used to generate an expanded lexicon from some initial lexicon. It is this expanded lexicon which is used by the chart parser.

Like lexical entries, the edges of the chart are also stored as unit clauses. As new edges are generated during the processing of some current edge, they are placed into a *processing stack* until the processing of the current edge has been completed. Since we use a *stack* instead of a *queue*, we get *depth-first* instead of *breadth-first* behaviour (Thompson, 1981). Just as we use the Prolog database to store our chart, we can also use it to store our processing queue. Each element **E** is stored as the single argument of the unit clause *stack* (ie. stack(E)).

```
parse(FA_Spec) :-
   readline(Sentence),                          % read sentence
   init(Sentence),                              % initialize the processing stack
   pop(Edge),                                   % get next edge from stack (and add it to chart)
   process(Edge, Sentence, FA_Spec).            % process current edge

parse(_).                                       % we've found all parses

process(edge(FA_Spec,Pend,Constits,[],Sentence,[],Number), Sentence, FA_Spec) :-
   phon_check(FA_Spec, Sentence, []),           % compare phonology of FA_Spec edge span
   eval_pend(Pend),                             % evaluate all pending constraints
   writesem(FA_Spec),                           % display semantic attribute of FA_Spec
   fail.                                        % try to find another parse

process(Edge, _, _) :-
   fundamental_rule(Edge),                      % apply fundamental rule to current edge
   rule_invocation(Edge),                       % attempt rule invocation
   fail.                                        % try next edge
```

Figure 4-8: The Chart Parser.

The Prolog parser, which is shown in Figure 4-8, attempts to find an FA specification that will unify with the "start symbol" FA specification given in *FA_Spec*. After reading in a sentence (readline) and initializing the processing stack (init), the parser gets an edge from the stack (pop) for processing. This processing of an edge entails finding other edges in the chart/database with which it can be combined (fundamental_rule) and checking to see if this edge triggers the introduction of new edges into the processing stack (rule_invocation).

Recall that a successful parse is reflected by the presence of an inactive edge whose FA specification unifies with the start symbol (FA_Spec). This edge must also span the entire input sentence (ie. its starting position must correspond to *Sentence* and its ending position must be the empty list). The root-sign of the FA specification of the edge must have a phonology that matches the input (phon_check). Furthermore, any constraints associated with the edge must be satisfiable (eval_pend). If an edge meets all of these criteria, then the parser displays the InL formula associated with the root-sign of the FA specification.

4.3.1. The Fundamental Rule

Recall that the fundamental rule states that a new edge is created when an active edge meets an inactive edge and the category of the inactive edge corresponds to the current expectation of the active edge. In TUG we are dealing with FA specifications instead of categories, so our fundamental rule will unify the FA specification associated with the inactive edge with the FA specification appearing as the current expectation of the active edge. The actual Prolog code associated with application of the fundamental rule of active chart parsing of TUG is introduced in Figure 4-9.

```
% fundamental_rule(FA_Specification, Constraints, Constituents, Expectations, LeftPos, RightPos, Num)

fundamental_rule(edge(FA_SpecI/[], ConI, _, [], PI1, PI2, I)) :-          % inactive edge
  edge(FA_Spec, ConA, _, [Comp|Ex], PA1, PA2, A),   % active edge
  meet(PI1, PI2, PA1, PA2, P1, P2),                 % do the edges meet?
  tree_unify(Comp, FA_SpecI/[]),                    % if so, unify expectation with inactive FA spec
  phon_check(FA_Spec, P1, P2),                      % compare phonology to edge span
  check_constraints(ConI, ConA, Con),               % evaluate any pending constraints
  push(edge(FA_Spec,Con,[A,I],Ex,P1,P2,_)),         % push new edge onto stack
  fail.                                             % look for other active edges

fundamental_rule(edge(FA_Spec, ConA, _, [Comp|Ex], PA1, PA2, A)) :-       % active edge
  edge(FA_SpecI, ConI, _, [], PI1, PI2, I),         % inactive edge
  meet(PI1, PI2, PA1, PA2, P1, P2),
  tree_unify(Comp, FA_SpecI),
  phon_check(FA_Spec, P1, P2),
  check_constraints(ConI, ConA, Con),
  push(edge(FA_Spec,Con,[A,I],Ex,P1,P2,_)),
  fail.                                             % look for other inactive edges

fundamental_rule(_).                                % when we've run out of edges

% meet(LeftPos1, RightPos1, LeftPos2, RightPos2, ResultLeftPos, ResultRightPos)

meet(L, P, P, R, L, R) :- P\==L, !.                 % 1st edge meets 2nd edge
meet(P, R, L, P, L, R) :- P\==L.                    % 2nd edge meets 1st edge

% tree_unify(Tree1, Tree2)

tree_unify(X,X).
```

Figure 4-9: Fundamental Rule.

The predicate *fundamental_rule* is applied to edges obtained from the processing stack as was illustrated in Figure 4-8. If the edge currently being processed is inactive (first clause of the definition), then we need to find an active edge that meets it. When processing active edges (second clause), we need to find inactive edges. Note that the unification of the auxiliary tree with the expectation of the primary tree is performed only if the two edges meet; the potentially computationally expensive tree unification is not performed on just any pair of active and inactive edges. Since the trees are represented as terms, this unification can be defined totally in terms of Prolog's primitive term unification operation. For grammar development purposes, it is often desirable to examine this unification process, though;

hence the definition of a separate relation.

As with HPSG rules, the TUG rule does not contain any information about the order of constituents in a sentence. All it states is that a primary tree combines with an auxiliary tree that can unify with the subtree associated with the first element of the auxiliary list of the primary tree. Conceivably, we might want to try combining any two FA specifications associated with some part of the sentence under analysis. This would eventually lead us to obtaining an FA specification whose root-node phonology matched the input sentence, but it would also lead to many other FA specifications which would not play a role in obtaining the final FA specification. The parsing process can be made much more efficient by only considering FA specifications associated with constituents adjacent to the region associated with the primary tree. Thus, the definition of *meets* in Figure 4-9 requires the active and inactive edges to be adjacent. Furthermore, for edges associated with constituents that span no constituents (ie. traces), the definition ensures that such edges can only be met in one direction (thus preventing duplicate analyses involving traces).

For most of the lexical entries introduced in section 3, the auxiliary lists are ordered so that the primary tree and auxiliary tree would be adjacent during the parsing process. The actual ordering of the elements in the auxiliary list does not really matter in our grammar; the same resulting FA specifications are obtained regardless.[3] We would only need to modify the auxiliary list for *promises*, so that it will combine with the object before the complement. In general, we will only have trouble constructing auxiliary lists that satisfy this *meets* criterion if we are dealing with FA specifications for discontinuous constituents.[4]

Phon_check ensures that the phonology attribute is consistent with the region spanned by the edge. This is necessary even with the *meets* restriction described above. Consider an attempted parse of the ungrammatical sentence *Walks Mary*. In this sentence, the primary tree associated with *Walks* is adjacent to the auxiliary tree for *Mary* and thus they can be combined to yield an FA specification for *Mary walks*. However, this FA specification does not correspond to the input sentence. The *phon_check* will rule out this analysis.

Finally, if FA specification resulting from the primary and auxiliary trees has satisfied all the criteria mentioned so far, any pending constraints that can be evaluated are checked. For example, a potential FA specification for *woman that loves himself* will be ruled out when the constraint on the reflexive attribute of the sign associated with *himself* is evaluated as part of the *check_constraints* relation. We will be discussing these constraints in more detail shortly.

---

[3]Note though that the order is relevant to our lexical rule for null subject instantiation. It requires the last element of the auxiliary list to correspond to the subtree dominating the subject. An alternative grammar could be constructed which would not require this lexical rule.

[4]TUG, like Lexicalized Tree Adjoining Grammar (Abeillé and Schabes, 1989), allows lexical entries to be proposed for discontinuous constituents. Examples of such lexical entries are introduced in the discussion of cleft-constructions in (Popowich, 1988).

The new edge resulting from a successful application of the fundamental rule will have the same starting position as the active edge and the same ending position as the inactive edge. Its expectations will be the remaining expectations from the active edge. The FA specification associated with the new edge will be a more instantiated version of that of the active edge. The active and inactive edges are not modified by fundamental rule application — the *fail* contained in the definition forces all modifications to be revoked after a copy of the new edge has been pushed onto the processing stack.

In our grammar, we require auxiliary trees to have empty auxiliary lists. Inactive edges with nonempty auxiliary lists can thus be ignored in fundamental rule application. Since they do not play a role in fundamental rule application there is no need to store inactive edges with nonempty auxiliary lists in chart. However, such edges will be subject to *rule invocation* and will thus cause the introduction of active edges.

4.3.2. Rule Invocation

Grammar rules are used to generate expectations for inactive edges that have nonempty auxiliary lists. The actual format of the rule used in the grammar is shown in (31).

(31) P/Rest --> P/[&A|Rest], A/[].

It states that the auxiliary tree *A* with an empty auxiliary list is unified with the structure pointed to by the first element of the auxiliary list of the primary tree. The ampersand in (31) is interpreted as referring to the structure referenced by the path specification contained in the auxiliary list. If our auxiliary lists contained the actual structures instead of pointers to the structures, the rule in (31) would be written as shown in (32).

(32) P/Rest --> P/[A|Rest], A/[].

Rules are converted into an internal format consisting of a two place predicate in which the first argument corresponds to the left hand side of the rule and the second argument to a list of the right hand side elements. So the rule introduced in (31) would be stored as shown in (33).

(33) rule(P/Rest, [P/[&A|Rest], A/[]]).

Although we have proposed a grammar which contains only one grammar rule, *rule_invocation*, as defined in Figure 4-10, allows grammars containing more than one rule. Furthermore, it can handle grammar rules like (32) in which the actual structures appear in the auxiliary list instead of the binary path specifications, and rules like (34) which contain more than one auxiliary tree.

(34) P/Rest --> P/[&A1,&A2|Rest], A1/[], A2/[].

The first clause of the *rule_invocation* definition can apply only to inactive edges, that is, edges having the empty list *[ ]* of expectations. The FA specification of this edge is a tree *Prim* with auxiliary list *[Ptr|Ptrs]*. A grammar rule is then invoked with *Prim* as the primary tree. Assuming the single grammar rule introduced in (31), this will result in *Aux* being instantiated to *[&A|Rest]*, *Comp* to *[A/[]]*, and *Mother* to *Prim/Rest*. The call to *select_tree* will result in *A* being unified with the subtree of *Prim* described by the

```
% rule_invocation( Inactive_Edge )

rule_invocation( edge(Prim/[Ptr|Ptrs], Pend, _, [], P1, P2, _) ) :-
  rule(Mother,[Prim/Aux | Comp]),                  % select a grammar rule
  select_tree(Prim, Aux, [Ptr|Ptrs]),              % select the appropriate subtree
  push( edge(Mother, Pend, [], Comp, P1, P2, _) ), % new edge with "Comp" as its expectations
  fail.                                            % try to find another grammar rule

rule_invocation(_).                                % no more grammar rules

% select_tree(Tree, Auxiliary_List_from_Rule, List_of_Binary_Path_Specifications)
%    match auxiliary list from rule with that of primary tree

select_tree(_, X, X) :- !.
select_tree(Tree, [Aux|Rest1], [Pnt|Rest2]) :-
  select_tree1(Tree, Aux, Pnt),
  select_tree(Tree, Rest1, Rest2).

% select_tree1(Tree, Element_from_Auxiliary_List, Binary_Path_Specification)

select_tree1(_, C, C) :- !.                        % when actual structures are used instead of paths
select_tree1(Tree, &Tree, []) :- !.                % select root
select_tree1(tree(_,F,_), &S, [f|Rest]) :- !, select_tree1(F, &S, Rest).          % select functor
select_tree1(tree(_,_,A), &S, [a|Rest]) :- !, select_tree1(A, &S, Rest).          % select argument
```

Figure 4-10: Rule Invocation.

binary path specification *Ptr*, and will result in the unification of *Rest* and *Ptrs*. After this unification is performed the FA specification corresponding to the auxiliary tree (*A/[]*) is used to form the expectation list of the new active edge. This new edge will have the same starting position (*P1*), ending position (*P2*) and constraint list (*Pend*) as the initial edge. Its FA specification will consist of *Prim* with auxiliary list *Ptrs*. Note that since the expectation contains some subtree of *Prim*, some future application of the fundamental rule will result in the expectation (and some subtree of *Prim*) being unified with some other FA specification.

4.3.3. Pending Constraints

In our chart parser, we have adopted a bottom-up processing strategy. One consequence of this decision is that we need to introduce edges for traces at each position in the input sentence during initialization. Another consequence is that it complicates the distribution of slash and R-antecedent information (i.e., environmental information as discussed in §2.1) that needs to be distributed in a top-down manner. For instance, the constraint *[...X|_]* associated with the lexical entry of the reflexive pronoun requires *X* to be an element of a list that does not become fully instantiated until later in the parse. The use of the freeze predicate which is available in certain Prologs allows evaluation of such constraints to be deferred. However, since this predicate is not universally available, we need to incorporate some mechanism for deferring evaluation of such constraints.

Since we are assuming a slash list containing at most one element, we can implement the C-SL relationship efficiently by treating the slash attribute as a single variable that is shared by the root signs and its functor and argument signs. We will allow the slash attribute to either take a *sign* value or an uninstantiated value, and we will interpret the uninstantiated value as nil. So if a functor and argument have nil slash values (variables), then the root-sign will also have a nil value. Similarly, if both of them have non-nil values, then they are required to be the same sign. If a functor introduces some non-nil slash value, it will be passed up onto the root-sign (with the side-effect that it will also appear on the argument-sign). We get a similar relationship when an argument introduces a non-nil value. This side-effect does not matter since we are only concerned with passing the slash value up the tree. We can

then have constraints associated with signs further up the tree that will test the value of the slash attribute. For example, if we are to parse a declarative sentence, then the slash value of the root-sign of its FA specification must be nil. So after parsing, we test to see if the slash value is a variable. If it is not, then some constituent has introduced a trace somewhere down the tree so we disallow the sentence. Similarly, in a relative clause construction, the sentence must have a sign corresponding to the trace as its slash value. If a test of this value shows it to be uninstantiated, then we have not introduced a trace lower in the tree. This method works due to our bottom-up parsing strategy.

```
edge(
  tree(
    sign(A-that-B,syn(cat(+n,-v,1),C,slash(D)),...),
    tree(sign(A,syn(cat(+n,-v,1),C,slash(D)),...),O,P),
    tree(
      sign(that-B,syn(cat(-n,+v,2),fin,slash(D)),...),
      tree(sign(B,syn(cat(-n,+v,2),fin,R),...),U,V),
      tree(sign(that,...),nil,nil)))/[[f]],
  [freeze(R, eq(sign([],syn(cat(+n,-v,2)...),...),R)),
   freeze(slash(D), nil(slash(D)))],
  [],
  [tree(sign(B,syn(cat(-n,+v,2),fin,R),...),U,V)/[]],
  [that,mary,loves,'.'],
  [mary,loves,'.'],
  30).
```

Figure 4-11: Edge with Constraints.

To illustrate how these constraints are treated in the chart, consider the active edge shown in Figure 4-11 which is associated with the word *loves* in the phrase *that Mary loves*. The FA specification contained in the first argument of *edge* has been abbreviated as reflected by the ellipsis symbols. There are two restrictions contained in its list of pending constraints. The first requires the variable **R**, when instantiated to some slash value, to be equal to the sign associated with a trace; it will be required to be the same as *slash(sign([],syn(cat(+n,-v,2)...))*. So if **R** becomes instantiated to something other than this value, or if it never becomes instantiated, this constraint will be violated and the constructions of the associated new edge will not succeed. The second restriction will require **D** to be uninstantiated; instantiation of **D** to some value will violate the constraint. The edge introduced in Figure 4-11 possesses an empty constituent list, and contains a single expectation (shown in italics) in its expectation list, thus being an active edge. Its starting position is to the left of the word *that* in the sentence, with its ending position being to the left of *Mary*. Finally, the number appearing in the final argument position of *edge* is the edge number.

Although we have talked about these constraints associated with reflexive pronouns and unbounded dependencies, we have not said much about how they are specified in FA specifications. Recall that we introduced the notation $[...f|_]_A$ in Figure 3-9 to represent a restriction *[...f|_]* applying to some attribute *A*. We will use this same subscripting notation to introduce constraints on other forms of information. So to constrain the slash list of a declarative sentence in a discourse (disc) to be nil, we will use the notation $nil(SL)_{SL}$ as illustrated in the lexical entry for the period shown in Figure 4-12. Note that multiple sentence discourses would be handled by a similar lexical entry for a period which, aside from an argument corresponding to a sentence, would have an additional argument corresponding to a discourse.

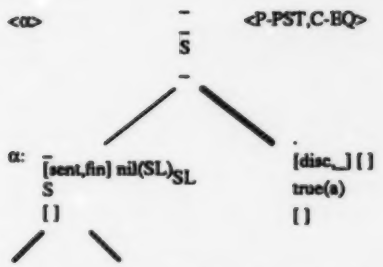

Figure 4-12: Lexical Entry for a Single Sentence Discourse.

5. Some Examples

Now that we have outlined how the parser works, we will give an example of an actual derivation of a simple sentence. Before we do this, we should state that our start symbol consists of an FA specification whose root-sign is associated with a discourse and thus possesses *disc* as its syntactic category. The grammar consists of 44 lexical entries: 17 verbs, 7 proper nouns including reflexives and the trace, 5 nouns, 4 prepositions, 3 adjectives, 2 articles, 2 relative pronouns and 4 entries for punctuation and affixes. Seventeen additional lexical entries are created by the lexical rule for Null Subject Instantiation.

```
tug> parse.
?john walks.

Sentence:  john walks .

To chart 7 spanning [john]
To chart 6 spanning []                          % trace preceding "john"
To chart 5 spanning [walks]                     % empty subject version
Process primary 4 spanning [walks]
Invocation of 8 from 4                          % active edge created from auxlist of edge 4
To chart 3 spanning []                          % trace preceding "walks"
Process primary 2 spanning [.]
Invocation of 9 from 2
To chart 1 spanning []                          % trace preceding "."
To chart 9 spanning [.]
Active 9 meets inactive 5 yielding 10
To chart 8 spanning [walks]
Active 8 meets inactive 7 yielding 11           % "walks" meets "john"
Active 8 meets inactive 1 yielding 12
To chart 12 spanning [walks]
To chart 11 spanning [john,walks]
Inactive 11 meets active 9 yielding 13          % "john walks" meets "."
To chart 10 spanning [walks,.]
To chart 13 spanning [john,walks,.]

[john(m1),walk(e2,m1)]
Time: 0.3 secs.
```

Figure 5-1: Trace of Successful Parse.

A trace of the chart parse of the sentence *John walks* is shown in Figure 5-1. User input is shown in italic, with bold font being used to highlight the operations that play a role in the successful parse. Observe that the period is included as one of the input symbols. Although thirteen edges are processed, not all of them are added to the chart.

The analysis of (35), a more complex sentence, requires the processing of 121 edges.

(35) Every woman loves a picture of herself that mary loves.

However, only 97 of these are actually added to the chart. Many of these edges are associated with partial analyses involving potential trace constituents. Due to our bottom-up processing strategy, edges associated with traces are introduced following every word in the sentence. The abbreviated InL

```
?every woman loves a picture of herself that mary loves.

Sentence:  every woman loves a picture of herself that mary loves .

[[woman(f1)]-->[[[[n2,[mary(f3),[n2,love(s4,f3,n2)]]],[[f3,of(n2,f3)],picture(n2)]]],love(s5,f1,n2)]]
Time: 6.538 secs.


[[woman(f6)]-->[[[[n7,[mary(f8),[n7,love(s9,f8,n7)]]],[[f6,of(n7,f6)],picture(n7)]]],love(s10,f6,n7)]]
Time: 6.8 secs.
```

Figure 5-2: Analysis of an Ambiguous Sentence.

translations of (35) are shown in Figure 5-2. InL formulae of the form *[x]true(x)* have been abbreviated simply as *x*. The results were obtained on a Sun SPARCstation running Quintus Prolog. The times shown are total times, so only an additional 0.2 seconds was required to obtain the additional parse in Figure 5-2. The program also runs under C-Prolog, and Sicstus Prolog on SUNs, and under Arity Prolog on IBM-PC compatibles.

Finally, consider the sentences introduced in (36) and (37).

(36) John persuades himself to love Mary's picture of herself.
(37) John persuades himself to love Mary's picture of himself.

When the reflexive is used anaphorically and not emphatically, the first sentence is grammatical while the second is not. In looking for all parses of the first sentence, the program requires 11 seconds to processes 152 edges, 113 of which are placed into the chart. The attempt to parse the second sentence requires 136 edges to be processed, 100 of which are added to the chart. Note that the actual input to the parser will be a list of the form shown in (38), so many of the edges will be associated with analyzing the first six list elements as a sentence.

(38) [john, persuades, himself, to, love, mary, 's, picture, of, himself]

6. Discussion and Conclusions

We have illustrated how a grammar consisting of a single grammar rule and a selection of lexical entries can be used to process English sentences containing reflexives, proper names, indefinite articles, possessives, unbounded dependencies, and controlled complements. For the lexical entries that have been introduced, the tree structures are for the most part based on categorial grammar derivation trees. Although argument-signs are arranged in order of semantic obliqueness, there is some disagreement on the actual ordering of semantic arguments. This issue is discussed by Dowty (1982).

R-antecedent information associated with both subject and non-subject antecedents is combined into the reflexive attribute of signs allowing a single lexical entry for the reflexive pronoun to be used for all of these cases. The reflexive attribute of signs has a status like the other attributes of the sign — no special treatment is awarded to R-antecedent information. Just like other forms of information, it can be used to prevent analyses for linguistic expressions that may be syntactically well-formed but which violate

anaphoric principles. If the reflexive attribute of a sign corresponding to a reflexive pronoun does not contain a variable of the appropriate sort for the antecedent, then there is no FA specification for the linguistic expression. The information contained in the reflexive attributes of signs is present in case it is ever needed. In this way, when a reflexive pronoun is encountered its potential antecedents are locally accessible — no complicated procedure is required to search for antecedents. The amount of information contained in reflexive attributes is small, and it is built up according to very simple and well founded relations (which are embodied in the reflexive templates). So the "space" and "effort" associated with this information is minimal. For signs associated with linguistic expressions that are not reflexive pronouns, although this information is not "used" (i.e., there are no restrictions on the reflexive attribute) it will be relevant to the distribution of reflexive pronouns contained within the expression. The R-antecedent information is present so that anaphora resolution can be stated in very simple terms.

Many constructions rely on the presence of signs with a null phonology ε. For instance, infinitival verb phrases are treated as sentences possessing null phonology subjects and the traces involved in unbounded dependencies have signs with null phonology associated with them. The presence of null phonology signs does not require us to propose lexical entries for null lexical items as we have illustrated in our treatment of control. However, we have also seen in our treatment of unbounded dependencues that such lexical entries can be handled by our parser.[5]

Various linguistic generalizations, like those associated with reflexivization, are captured in the templates and lexical rules of the lexicon. The use of templates and lexical rules allows the lexicon to be stated more concisely with less duplication of information. Since the lexicon is expanded before the actual parsing begins, there is no need to complicate our chart parser with mechanisms for processing templates and lexical rules. Many details concerning how lexical rules have been glossed over since they are not relevant to our small lexicon. In a larger lexicon, problems relating to the order of lexical rule application during the lexicon expansion stage would need to be addressed. Lexical rules also have the undesirable property of having the potential to result in an infinite lexicon. Therefore care must be taken to avoid such consequences when both developing and processing a grammar.

At the centre of our grammar processor lies a chart parser for the TUG formalism. It manipulates FA specifications that have been converted into Prolog terms. Some of the restrictions associated with FA specifications (namely, those associated with unbounded dependencies and reflexivization) cannot be directly incorporated into these terms, so the chart parser treats them as constraints on specific values within the terms. Some heuristics related to the form of the TUG grammar rule have been incorporated into the parser to decrease the size of the chart and thus improve efficiency. Additional heuristics (perhaps in the form of top-down information filtering) could be incorporated into the parser to avoid proposing trace consituents at every word boundary in the input.

---

[5]A proposal for unbounded dependency constructions that does not require the introducion of a lexical entry for trace in provided by Popowich (1988).

Although only one grammar rule is required, this is at the cost of introducing additional lexical entries. Specifically, the lexical rule for null subject instantiation doubles the number of lexical rules for verbs. This proliferation of lexical entries is a problem for such a highly lexical formalism like TUG.[6] In order to examine this problem and the problems associated with lexical rules in more detail, a grammar development system for constructing, manipulating, and displaying the templates and FA specifications of TUG is currently under development. With this system, we will be able to investigate grammars providing alternative analyses of verb phrases and noun phrases that would have smaller lexicons. Also, since our current grammar embodies only the *necessary* constraints for the constructions being analyzed (there may be additional constraints on sentences which might make them ungrammatical), an extended grammar (developed with a grammar development system) could incorporate additional constraints.

Acknowledgements

Many thanks to Carl Vogel and Dan Fass for their comments on, and discussions about, earlier drafts of this paper. Special thanks to Elisabet Engdahl and Ewan Klein for discussions concerning my Ph.D. dissertation which formed the basis for the material in sections 2 and 3.

This research is supported by a research fellowship from the Advanced Systems Institute of British Columbia and by the Natural Sciences and Engineering Research Council of Canada under Grant No. OGP0041910.

References

Abeillé, Anne, and Yves Schabes. (1989). Parsing Idioms in Lexicalized TAGs. In: *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics*. Manchester, England.

Bach, Emmon. (1979). Control in Montague Grammar. *Linguistic Inquiry, 10*, 515-531.

Bach, Emmon, and Barbara Partee. (1980). Anaphora and Semantic Structure. In C. Masek, P. Hendrick and M. Miller (Eds.), *Papers from the Parasession on Language and Behavior at the 17th Regional Meeting of the Chicago Linguistics Society*. Chicago, IL.

Calder, Jo, Ewan Klein, and Henk Zeevat. (1988). Unification Categorial Grammar: A Concise, Extendable Grammar for Natural Language Processing. In: *12th International Conference on Computational Linguistics*. Budapest, Hungary.

Calder, Jo, Marc Moens, and Henk Zeevat. (1986). *A UCG Interpreter* (Deliverable T2.6). ESPRIT PROJECT 393 ACORD, Centre for Cognitive Science, Edinburgh, Scotland.

Chomsky, Noam. (1970). Remarks on Nominalization. In R. Jacobs and P. Rosenbaum (Eds.), *Readings in English Transformational Grammar*. Ginn and Co., Waltham, MA.

Chomsky, Noam. (1973). Conditions on Transformations. In S. Anderson and P. Kiparski (Eds.), *A Festschrift for Morris Halle*. Holt, Rinehart and Winston, New York, NY.

Chomsky, Noam. (1975). *Reflections on Language*. Pantheon Books, New York, NY.

Clocksin, William, and Chris Mellish. (1981). *Programming in Prolog*. Springer-Verlag, Berlin, Heidelberg, West Germany.

Dowty, David. (1982). Grammatical Relations and Montague Grammar. In P. Jacobson and G. Pullum

[6] The relationship between TUG and other highly lexical formalisms is discussed in (Popowich, 1988, Popowich, 1989a).

(Eds.), *The Nature of Syntactic Representation*. D. Reidel, Dordrecht, Holland.

Engdahl, Elisabet. (1983). Parasitic Gaps. *Linguistics and Philosophy*, 6, 5-34.

Flickinger, Dan. (1987). *Lexical Rules in the Hierarchical Lexicon*. Doctoral dissertation, Stanford University, CA.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullum, and Ivan Sag. (1985). *Generalized Phrase Structure Grammar*. Basil Blackwell, London, England.

Johnson, Mark, and Ewan Klein. (1986). Discourse, Anaphora and Parsing. In: *11th International Conference on Computational Linguistics*. Bonn University, West Germany.

Kameyama, M. (1985). *Zero Anaphora: The Case of Japanese*. Doctoral dissertation, Department of Linguistics, Stanford University, Stanford, CA.

Kamp, Hans. (1981). A Theory of Truth and Semantic Representation. In J. Groenendijk, T. Janssen, and M. Stokhof (Eds.), *Formal Methods in the Study of Language*. Mathematical Centre Tracts, Amsterdam, Holland.

Kaplan, Ron. (1973). A General Syntactic Processor. In R. Rustin (Ed.), *Natural Language Processing*. Algorithmics Press, New York, NY.

Kaplan, Ron, and Joan Bresnan. (1982). Lexical-Functional Grammar: A Formal System for Grammatical Representation. In J. Bresnan (Ed.), *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA.

Kay, Martin. (1973). The MIND System. In R. Rustin (Ed.), *Natural Language Processing*. Algorithmics Press, New York, NY.

Keenan, Edward. (1974). The Functional Principle: Generalizing the Notion of 'Subject of'. In M. La Galy, R. Fox, and A. Bruck (Eds.), *Papers from the 10th Regional Meeting of the Chicago Linguistics Society*. Chicago, IL.

Keenan, Edward, and Leonard Faltz. (1978). *Logical Types for Natural Language*. UCLA Occasional Papers in Linguistics, No. 3.

Klein, Ewan. (1987). DRT in Unification Categorial Grammar. In B. Lowden (Ed.), *Proceedings of the Alvey Workshop on Formal Semantics in Natural Language Processing, March 6 1987*. University of Essex, England.

Oehrle, Richard, Emmon Bach, and Deirdre Wheeler. (1988). *Categorial Grammars and Natural Language Structures*. D. Reidel, Dordrecht, Holland.

Partee, Barbara. (1984). Compositionality. In F. Landman and F. Veltman (Eds.), *Varieties of Formal Semantics: Proceedings of the Fourth Amsterdam Colloquium, September 1982*. Dordrecht, Holland.

Pollard, Carl. (1984). *Generalized Phrase Structure Grammars, Head Grammars, and Natural Languages*. Doctoral dissertation, Stanford University, CA.

Pollard, Carl, and Ivan Sag. (1983). Reflexives and Reciprocals in English: An Alternative to the Binding Theory. In M. Barlow, D. Flickinger, and M. Westcoat (Eds.), *Proceedings of the 2nd West Coast Conference on Formal Linguistics*. Stanford, CA.

Pollard, Carl, and Ivan Sag. (1987). *Information-Based Syntax and Semantics, Volume 1: Fundamentals*. Centre for the Study of Language and Information, Stanford University, CA.

Pollard, Carl, and Ivan Sag. (1989). Argument Structure and Binding in English. Unpublished manuscript, Centre for the Study of Language and Information, Stanford University, CA.

Popowich, Fred. (1988). *Reflexives and Tree Unification Grammar*. Doctoral dissertation, Centre for Cognitive Science, University of Edinburgh, Edinburgh, Scotland.

Popowich, Fred. (1989a). Tree Unification Grammar. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*. Vancouver, B.C.

Popowich, Fred. (1989b). *Reflexives in Head-Driven Phrase Structure Grammar* (CSS/LCCR TR 89-13). Simon Fraser University, Burnaby, B.C.

Proudian, Derek and Carl Pollard. (1985). Parsing Head-Driven Phrase Structure Grammar. In:

Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics. University of Chicago, Chicago, IL.

Reinhart, Tanya. (1981). Definite NP Anaphora and C-Command Domains. *Linguistic Inquiry, 12,* 605-635.

Seiffert, Roland. (1987). *Chart-Parsing of Unification-Based Grammars with ID/LP-Rules* (LILOG-REPORT 22). IBM Deutschland GmbH, LILOG, Stuttgart, West Germany.

Shieber, Stuart. (1985). Using Restriction to Extend Parsing Algorithms for Complex-Feature-Based Formalisms. In: *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics.* University of Chicago, Chicago, IL.

Shieber, Stuart. (1986). *An Introduction to Unification-Based Approaches to Grammar.* The University of Chicago Press, Chicago, IL.

Shieber, Stuart, Hans Uszkoreit, Fernando Pereira, Jane Robinson, and M. Tyson. (1983). The Formalism and Implementation of PATR-II. In B. Grosz and M. Stickel (Eds.), *Research on Interactive Acquisition and Use of Knowledge.* SRI International, Menlo Park, CA.

Thompson, Henry. (1981). Chart Parsing and Rule Schemata in PSG. In: *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics.* Stanford University, Stanford, CA.

Uszkoreit, Hans. (1986). Categorial Unification Grammars. In: *11th International Conference on Computational Linguistics.* Bonn University, West Germany.

Zeevat, Henk. (1988). Combining Categorial Grammar and Unification. In U. Reyle and C. Rohrer (Eds.), *Natural Language Parsing and Linguistic Theory.* D. Reidel, Dordrecht, Holland.

Zeevat, Henk, Ewan Klein, and Jo Calder. (1987). An Introduction to Unification Categorial Grammar. In N. Haddock, E. Klein, and G. Morrill (Eds.), *Edinburgh Working Papers in Cognitive Science, Vol.1: Categorial Grammar, Unification Grammar, and Parsing.* Centre for Cognitive Science, University of Edinburgh, Scotland.

Centre for Systems Science

• Simon Fraser University established the **Centre for Systems Science** as a provincial centre for advanced study. The objective of CSS is twofold: coordinate and combine advanced systems research at SFU, and encourage application of research results within industry in order to further the technological development of British Columbia.

• CSS has assembled three research groups to streamline its efforts. The orientation of these groups is as follows: intelligent systems, microelectronics, and computer and communications systems. Because of shared research interests, this latter group merges with LCCR (**Laboratory for Computer and Communications Research**) for the publication of technical reports.

• CSS's three groups involve over 80 skilled faculty researchers principally from SFU's Schools of Engineering Science and Computing Science, but also draw substantially on Mathematics, Physics, Geography, Philosophy, Linguistics, Psychology, Communication, and Kinesiology.